

glossaries.sty v2.07: L^AT_EX 2_ε Package to Assist Generating Glossaries

Nicola L.C. Talbot

School of Computing Sciences
University of East Anglia
Norwich, Norfolk
NR4 7TJ, United Kingdom.

<http://theoval.cmp.uea.ac.uk/~nlct/>

2010-07-10

This is the user manual for the glossaries package. Other documents accompanying this manual:

- “The glossaries package: a guide for beginners” ([glossariesbegin.pdf](#))
- “Upgrading from the glossary package to the glossaries package” ([glossary2glossaries.pdf](#))

See the file INSTALL for installation instructions. Related web resources:

- The glossaries FAQ¹
- Glossaries, Nomenclature, Lists of Symbols and Acronyms²

¹<http://theoval.cmp.uea.ac.uk/~nlct/latex/packages/faq/glossariesfaq.html>

²http://www.latex-community.org/index.php?option=com_content&view=article&id=263&Itemid=114

Contents

1	Introduction	4
1.1	Sample Documents	5
1.2	Multi-Lingual Support	12
1.2.1	Changing the Fixed Names	12
1.3	Generating the Associated Glossary Files	15
1.3.1	Using the makeglossaries Perl Script	17
1.3.2	Using xindy explicitly	17
1.3.3	Using makeindex explicitly	18
1.3.4	Note to Front-End and Script Developers	19
1.4	Troubleshooting	20
2	Overview of Main User Commands	23
2.1	Package Options	23
2.2	Defining Glossary Entries	28
2.2.1	Plurals	32
2.2.2	Sub-Entries	33
2.2.3	Loading Entries From a File	35
2.3	Number lists	36
2.4	Links to Glossary Entries	37
2.4.1	Changing the format of the link text	45
2.4.2	Enabling and disabling hyperlinks to glossary entries	47
2.5	Adding an Entry to the Glossary Without Generating Text	47
2.6	Cross-Referencing Entries	48
2.7	Using Glossary Terms Without Links	49
2.8	Displaying a glossary	52
2.8.1	Changing the way the entry name appears in the glossary	54
2.8.2	Xindy	54
2.9	Defining New Glossaries	58
2.10	Acronyms	59
2.10.1	Defining A Custom Acronym Style	67
2.10.2	Upgrading From the glossary Package	70
2.11	Unsetting and Resetting Entry Flags	71
2.12	Glossary Styles	72
2.12.1	List Styles	74
2.12.2	Longtable Styles	75
2.12.3	Longtable Styles (Ragged Right)	77
2.12.4	Supertabular Styles	78
2.12.5	Supertabular Styles (Ragged Right)	80
2.12.6	Tree-Like Styles	81
2.13	Defining your own glossary style	82
2.13.1	Example: creating a completely new style	85
2.13.2	Example: creating a new glossary style based on an existing style	85

2.13.3 Example: creating a glossary style that uses the <code>user1, ..., user6</code> keys	86
2.14 Accessibility Support	87
3 Mfirstuc Package	87
4 Glossaries Documented Code	89
4.1 Package Definition	89
4.2 Package Options	89
4.3 Default values	99
4.4 Xindy	106
4.5 Loops and conditionals	113
4.6 Defining new glossaries	114
4.7 Defining new entries	116
4.8 Resetting and unsetting entry flags	126
4.9 Loading files containing glossary entries	127
4.10 Using glossary entries in the text	128
4.10.1 Links to glossary entries	129
4.10.2 Displaying entry details without adding information to the glossary	169
4.11 Adding an entry to the glossary without generating text	173
4.12 Creating associated files	174
4.13 Writing information to associated files	182
4.14 Glossary Entry Cross-References	183
4.15 Displaying the glossary	185
4.16 Acronyms	194
4.17 Predefined acronym styles	197
4.18 Predefined Glossary Styles	217
5 Mfirstuc Documented Code	217
6 Glossary Styles	219
6.1 Glossary hyper-navigation definitions (glossary-hypernav package)	219
6.2 List Style (glossary-list.sty)	221
6.3 Glossary Styles using longtable (the glossary-long package)	224
6.4 Glossary Styles using longtable (the glossary-longragged package)	229
6.5 Glossary Styles using supertabular environment (glossary-super package)	234
6.6 Glossary Styles using supertabular environment (glossary-superragged package)	240
6.7 Tree Styles (glossary-tree.sty)	246
7 Accessibilty Support (glossaries-accsupp Code)	253
7.1 Defining Replacement Text	254
7.2 Accessing Replacement Text	256
7.3 Displaying the Glossary	265

7.4 Acronyms	266
8 Multi-Lingual Support	269
8.1 Babel Captions	269
8.2 Polyglossia Captions	275
8.3 Brazilian Dictionary	278
8.4 Danish Dictionary	278
8.5 Dutch Dictionary	279
8.6 English Dictionary	279
8.7 French Dictionary	279
8.8 German Dictionary	280
8.9 Irish Dictionary	280
8.10 Italian Dictionary	280
8.11 Magyar Dictionary	280
8.12 Polish Dictionary	281
8.13 Serbian Dictionary	281
8.14 Spanish Dictionary	281
Index	283

1 Introduction

The `glossaries` package is provided to assist generating glossaries. It has a certain amount of flexibility, allowing the user to customize the format of the glossary and define multiple glossaries. It also supports acronyms and glossary styles that include symbols (in addition to a name and description) for glossary entries. There is provision for loading a database of glossary terms. Only those terms used³ in the document will be added to the glossary.

This package replaces the `glossary` package which is now obsolete. Please see the document “Upgrading from the `glossary` package to the `glossaries` package” ([glossary2glossaries.pdf](#)) for assistance in upgrading.

One of the strengths of this package is its flexibility, however the drawback of this is the necessity of having a large manual that can cover all the various settings. If you are daunted by the size of the manual, try starting off with the much shorter guide for beginners ([glossariesbegin.pdf](#)).

The `glossaries` package comes with a **Perl** script called `makeglossaries`. This provides a convenient interface to `makeindex` or `xindy`. It is strongly recommended that you use this script, but *it is not essential*. If you are reluctant to install Perl, or for any other reason you don’t want to use `makeglossaries`, you can call `makeindex` or `xindy` explicitly. See Section 1.3 for further details.

This manual is structured as follows:

³that is, if the term has been referenced using any of the commands described in Section 2.4, Section 2.5 or via `\glssee` (or the `see` key)

- Section 2 gives an overview of the main user commands and their syntax.
- Section 3 describes the associated `mfirstuc` package.
- Section 4 contains the documented source code for those who want to know more about how the package works. This describes more advanced commands, such as determining if an entry or a glossary exists and commands that iterate through defined terms or glossaries.
- Section 5 contains the documented code for the `mfirstuc` package.

The remainder of this introductory section covers the following:

- Section 1.1 lists the sample documents provided with this package.
- Section 1.2 provides information for users who wish to write in a language other than English.
- Section 1.3 describes how to use a post-processor to create the sorted glossaries for your document.
- Section 1.4 provides some assistance in the event that you encounter a problem.

1.1 Sample Documents

The `glossaries` package is provided with some sample documents that illustrate the various functions. These should be located in the `samples` subdirectory (folder) of the `glossaries` documentation directory. This location varies according to your operating system and \TeX distribution. You can use `texdoc` to locate the main `glossaries` documentation. For example, in a terminal or command prompt, type:

```
texdoc -l glossaries
```

This should display the full pathname of the file `glossaries.pdf`. View the contents of that directory and see if it contains the `samples` subdirectory.

If you can't find the sample files, they are available in the subdirectory `doc/latex/glossaries/samples/` in the `glossaries.tds.zip` archive which can be downloaded from [CTAN](#).

The sample documents are as follows:

minimalgls.tex This document is a minimal working example. You can test your installation using this file. To create the complete document you will need to do the following steps:

1. Run `minimalgls.tex` through \LaTeX either by typing

```
latex minimalgls
```

in a terminal or by using the relevant button or menu item in your text editor or front-end. This will create the required associated files but you will not see the glossary. If you use PDF \LaTeX you will also get warnings about non-existent references. These warnings may be ignored on the first run.

If you get a `Missing \begin{document}` error, then it's most likely that your version of `xkeyval` is out of date. Check the log file for a warning of that nature. If this is the case, you will need to update the `xkeyval` package.

2. Run `makeglossaries` on the document. This can be done on a terminal either by typing

```
makeglossaries minimalgls
```

or by typing

```
perl makeglossaries minimalgls
```

If your system doesn't recognise the command `perl` then it's likely you don't have Perl installed. In which case you will need to use `makeindex` directly. You can do this in a terminal by typing (all on one line):

```
makeindex -s minimalgls.ist -t minimalgls.glg -o minimalgls.gls  
minimalgls.glo
```

(See Section 1.3.3 for further details on using `makeindex` explicitly.)

Note that if you need to specify the full path and the path contains spaces, you will need to delimit the file names with the double-quote character.

3. Run `minimalgls.tex` through \LaTeX again (as step 1)

You should now have a complete document. The number following each entry in the glossary is the location number. By default, this is the page number where the entry was referenced.

sample4col.tex This document illustrates a four column glossary where the entries have a symbol in addition to the name and description. To create the complete document, you need to do:

```
latex sample4col  
makeglossaries sample4col  
latex sample4col
```

As before, if you don't have Perl installed, you will need to use `makeindex` directly instead of using `makeglossaries`. The vertical gap between entries is the gap created at the start of each group. This can be suppressed by redefining `\glsgroupskip` after the glossary style has been set:

```
\renewcommand*{\glsgroupskip}{}
```

sampleAcr.tex This document has some sample acronyms. It also adds the glossary to the table of contents, so an extra run through L^AT_EX is required to ensure the document is up to date:

```
latex sampleAcr
makeglossaries sampleAcr
latex sampleAcr
latex sampleAcr
```

sampleAcrDesc.tex This is similar to the previous example, except that the acronyms have an associated description. As with the previous example, the glossary is added to the table of contents, so an extra run through L^AT_EX is required:

```
latex sampleAcrDesc
makeglossaries sampleAcrDesc
latex sampleAcrDesc
latex sampleAcrDesc
```

sampleDesc.tex This is similar to the previous example, except that it defines the acronyms using `\newglossaryentry` instead of `\newacronym`. As with the previous example, the glossary is added to the table of contents, so an extra run through L^AT_EX is required:

```
latex sampleDesc
makeglossaries sampleDesc
latex sampleDesc
latex sampleDesc
```

sample-custom-acronym.tex This document illustrates how to define your own acronym style if the predefined styles don't suit your requirements.

```

latex sample-custom-acronym
makeglossaries sample-custom-acronym
latex sample-custom-acronym

```

sampleDB.tex This document illustrates how to load external files containing the glossary definitions. It also illustrates how to define a new glossary type. This document has the number list suppressed and uses `\glsaddall` to add all the entries to the glossaries without referencing each one explicitly. To create the document do:

```

latex sampleDB
makeglossaries sampleDB
latex sampleDB

```

The glossary definitions are stored in the accompanying files `database1.tex` and `database2.tex`. Note that if you don't have Perl installed, you will need to use `makeindex` twice instead of a single call to `makeglossaries`:

1. Create the main glossary:

```
makeindex -s sampleDB.ist -t sampleDB.glg -o sampleDB.gls sampleDB.glo
```

2. Create the secondary glossary:

```
makeindex -s sampleDB.ist -t sampleDB.nlg -o sampleDB.not sampleDB.ntn
```

sampleEq.tex This document illustrates how to change the location to something other than the page number. In this case, the `equation` counter is used since all glossary entries appear inside an `equation` environment. To create the document do:

```

latex sampleEq
makeglossaries sampleEq
latex sampleEq

```

sampleEqPg.tex This is similar to the previous example, but the number lists are a mixture of page numbers and equation numbers. This example adds the glossary to the table of contents, so an extra `LATEX` run is required:

```
latex sampleEqPg
```



```
makeglossaries sampleEqPg
latex sampleEqPg
latex sampleEqPg
```

sampleSec.tex This document also illustrates how to change the location to something other than the page number. In this case, the **section** counter is used. This example adds the glossary to the table of contents, so an extra \LaTeX run is required:

```
latex sampleSec
makeglossaries sampleSec
latex sampleSec
latex sampleSec
```

sampleNtn.tex This document illustrates how to create an additional glossary type. This example adds the glossary to the table of contents, so an extra \LaTeX run is required:

```
latex sampleNtn
makeglossaries sampleNtn
latex sampleNtn
latex sampleNtn
```

Note that if you don't have Perl installed, you will need to use **makeindex** twice instead of a single call to **makeglossaries**:

1. Create the main glossary:

```
makeindex -s sampleNtn.ist -t sampleNtn.glg -o sampleNtn.gls
sampleNtn.glo
```

2. Create the secondary glossary:

```
makeindex -s sampleNtn.ist -t sampleNtn.nlg -o sampleNtn.not
sampleNtn.ntn
```

sample.tex This document illustrates some of the basics, including how to create child entries that use the same name as the parent entry. This example adds the glossary to the table of contents, so an extra \LaTeX run is required:

```

latex sample
makeglossaries sample
latex sample
latex sample

```

You can see the difference between word and letter ordering if you substitute `order=word` with `order=letter`. (Note that this will only have an effect if you use `makeglossaries`. If you use `makeindex` explicitly, you will need to use the `-l` switch to indicate letter ordering.)

sampletree.tex This document illustrates a hierarchical glossary structure where child entries have different names to their corresponding parent entry. To create the document do:

```

latex sampletree
makeglossaries sampletree
latex sampletree

```

samplexdy.tex This document illustrates how to use the `glossaries` package with `xindy` instead of `makeindex`. The document uses UTF8 encoding (with the `inputenc` package). The encoding is picked up by `makeglossaries`. By default, this document will create a `xindy` style file called `samplexdy.xdy`, but if you uncomment the lines

```

\setStyleFile{samplexdy-mc}
\noist
\GlsSetXdyLanguage{}

```

it will set the style file to `samplexdy-mc.xdy` instead. This provides an additional letter group for entries starting with “Mc” or “Mac”. If you use `makeglossaries`, you don’t need to supply any additional information. If you don’t use `makeglossaries`, you will need to specify the required information. Note that if you set the style file to `samplexdy-mc.xdy` you must also specify `\noist`, otherwise the `glossaries` package will overwrite `samplexdy-mc.xdy` and you will lose the “Mc” letter group.

To create the document do:

```

latex samplexdy
makeglossaries samplexdy
latex samplexdy

```

If you don't have Perl installed, you will have to call `xindy` explicitly instead of using `makeglossaries`. If you are using the default style file `samplexdy.xdy`, then do (no line breaks):

```
xindy -L english -C utf8 -I xindy -M samplexdy -t samplexdy.glg  
-o samplexdy.gls samplexdy.glo
```

otherwise, if you are using `samplexdy-mc.xdy`, then do (no line breaks):

```
xindy -I xindy -M samplexdy-mc -t samplexdy.glg -o samplexdy.gls  
samplexdy.glo
```

sampleutf8.tex This is another example that uses `xindy`. Unlike `makeindex`, `xindy` can cope with accented or non-Latin characters. This document uses UTF8 encoding. To create the document do:

```
latex sampleutf8  
makeglossaries sampleutf8  
latex sampleutf8
```

If you don't have Perl installed, you will have to call `xindy` explicitly instead of using `makeglossaries` (no line breaks):

```
xindy -L english -C utf8 -I xindy -M sampleutf8 -t sampleutf8.glg  
-o sampleutf8.gls sampleutf8.glo
```

If you remove the `xindy` option from `sampleutf8.tex` and do:

```
latex sampleutf8  
makeglossaries sampleutf8  
latex sampleutf8
```

you will see that the entries that start with a non-Latin character now appear in the symbols group, and the word “manœuvre” is now after “manor” instead of before it. If you are unable to use `makeglossaries`, the call to `makeindex` is as follows (no line breaks):

```
makeindex -s sampleutf8.ist -t sampleutf8.glg -o sampleutf8.gls  
sampleutf8.glo
```

sampleaccsupp.tex This document uses the experimental `glossaries-accsupp` package. The symbol is set to the replacement text. Note that some PDF viewers don't use the accessibility support. Information about the `glossaries-accsupp` package can be found in Section 2.14.

1.2 Multi-Lingual Support

As from version 1.17, the `glossaries` package can now be used with `xindy` as well as `makeindex`. If you are writing in a language that uses accented characters or non-Latin characters it is recommended that you use `xindy` as `makeindex` is hard-coded for Latin languages. This means that you are not restricted to the A, ..., Z letter groups. If you want to use `xindy`, remember to use the `xindy` package option. For example:

```
\documentclass[frenchb]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{babel}
\usepackage[xindy]{glossaries}
```

If you use an accented or other expandable character at the start of an entry name, you must place it in a group, or it will cause a problem for commands that convert the first letter to uppercase (e.g. `\Gls`) due to expansion issues. For example:

```
\newglossaryentry{elite}{name={{é}lite},
description={select group or class}}
```

If you use the `inputenc` package, `makeglossaries` will pick up the encoding from the auxiliary file. If you use `xindy` explicitly instead of via `makeglossaries`, you may need to specify the encoding using the `-C` option. Read the `xindy` manual for further details.

1.2.1 Changing the Fixed Names

As from version 1.08, the `glossaries` package now has limited multi-lingual support, thanks to all the people who have sent me the relevant translations either via email or via `comp.text.tex`. However you must load `babel` or `polyglossia` *before* `glossaries` to enable this. Note that if `babel` is loaded and the `translator` package is detected on TeX's path, then the `translator` package will be loaded automatically. However, it may not pick up on the required languages so, if the predefined text is not translated, you may need to explicitly load the `translator` package with the required languages. For example:

```
\usepackage[spanish]{babel}
\usepackage[spanish]{translator}
\usepackage{glossaries}
```

Alternatively, specify the language as a class option rather than a package option. For example:

```
\documentclass[spanish]{report}

\usepackage{babel}
\usepackage{glossaries}
```

If you want to use `ngerman` or `german` instead of `babel`, you will need to include the `translator` package to provide the translations. For example:

```
\documentclass[ngerman]{article}
\usepackage{ngerman}
\usepackage{translator}
\usepackage{glossaries}
```

The languages are currently supported by the `glossaries` package are listed in [table 1](#).

Table 1: Supported Languages

Language	As from version
Brazilian Portuguese	1.17
Danish	1.08
Dutch	1.08
English	1.08
French	1.08
German	1.08
Irish	1.08
Italian	1.08
Hungarian	1.08
Polish	1.13
Serbian	2.06
Spanish	1.08

The language dependent commands and `translator` keys used by the `glossaries` package are listed in [table 2](#).

Due to the varied nature of `glossaries`, it's likely that the predefined translations may not be appropriate. If you are using the `babel` package and do not have the `translator` package installed, you need to be familiar with the advice given in <http://www.tex.ac.uk/cgi-bin/texfaq2html?label=latexwords>. If you have the `translator` package installed, then you can provide your own dictionary with the necessary modifications (using `\deftranslation`) and load it using `\usedictionary`. Note that the dictionaries are loaded at the beginning of the document, so it won't have any effect if you put `\deftranslation` in the preamble. It should be put in your personal dictionary instead. See the `translator` documentation for further details.

Table 2: Customised Text

Command Name	Translator Key Word	Purpose
<code>\glossaryname</code>	Glossary	Title of the main glossary.
<code>\acronymname</code>	Acronyms	Title of the list of acronyms (when used with package option <code>acronym</code>).
<code>\entryname</code>	Notation (glossaries)	Header for first column in the glossary (for 2, 3 or 4 column glossaries that support headers).
<code>\descriptionname</code>	Description (glossaries)	Header for second column in the glossary (for 2, 3 or 4 column glossaries that support headers).
<code>\symbolname</code>	Symbol (glossaries)	Header for symbol column in the glossary for glossary styles that support this option.
<code>\pagelistname</code>	Page List (glossaries)	Header for page list column in the glossary for glossaries that support this option.
<code>\glssymbolsgroupname</code>	Symbols (glossaries)	Header for symbols section of the glossary for glossary styles that support this option.
<code>\glsnumbersgroupname</code>	Numbers (glossaries)	Header for numbers section of the glossary for glossary styles that support this option.

If you are using `babel` and don't want to use the translator interface, you can suppress it using the package option `translate=false`, and either load `glossaries-babel` after `glossaries` or specify you're own translations. For example:

```
\documentclass[british]{article}

\usepackage{babel}
\usepackage[translate=false]{glossaries}
\usepackage{glossaries-babel}

or:

\documentclass[british]{article}

\usepackage{babel}
\usepackage[translate=false]{glossaries}

\addto\captionsbritish{%
  \renewcommand*{\glossaryname}{List of Terms}%
  \renewcommand*{\acronymname}{List of Acronyms}%
  \renewcommand*{\entryname}{Notation}%
  \renewcommand*{\descriptionname}{Description}%
  \renewcommand*{\symbolname}{Symbol}%
  \renewcommand*{\pagelistname}{Page List}%
  \renewcommand*{\glssymbolsgroupname}{Symbols}%
  \renewcommand*{\glsnumbersgroupname}{Numbers}%
}
```

If you are using `polyglossia` instead of `babel`, `glossaries-polyglossia` will automatically be loaded unless you specify the package option `translate=false`.

Note that `xindy` provides much better multi-lingual support than `makeindex`, so it's recommended that you use `xindy` if you have glossary entries that contain diacritics or non-Roman letters. See Section 2.8.2 for further details.

1.3 Generating the Associated Glossary Files

In order to generate a sorted glossary with compact location lists, it is necessary to use an external indexing application as an intermediate step. It is this application that creates the file containing the code that typesets the glossary. If this step is omitted, the glossaries will not appear in your document. The two indexing applications that are most commonly used with \LaTeX are `makeindex` and `xindy`. As from version 1.17, the `glossaries` package can be used with either of these applications. Previous versions were designed to be used with `makeindex` only. Note that `xindy` has much better multi-lingual support than `makeindex`, so `xindy` is recommended if you're not writing in English. Commands that only have an effect when `xindy` is used are described in Section 2.8.2.

The `glossaries` package comes with the Perl script `makeglossaries` which will run `makeindex` or `xindy` on all the glossary files using a customized style file (which is created by `\makeglossaries`). See Section 1.3.1 for further details.

Perl is stable, cross-platform, open source software that is used by a number of T_EX-related applications. Further information is available at <http://www.perl.org/about.html>. However, whilst it is strongly recommended that you use the `makeglossaries` script, it is possible to use the `glossaries` package without having Perl installed. In which case, if you have used the `xindy` package option, you will need to use `xindy` (see Section 1.3.2), otherwise you will need to use `makeindex` (see Section 1.3.3). Note that some commands and package options have no effect if you don't use `makeglossaries`. These are listed in table 3.

Note that if any of your entries use an entry that is not referenced outside the glossary, you will need to do an additional `makeglossaries`, `makeindex` or `xindy` run, as appropriate. For example, suppose you have defined the following entries:

```
\newglossaryentry{citrusfruit}{name={citrus fruit},
description={fruit of any citrus tree. (See also
\gls{orange})}}
```

```
\newglossaryentry{orange}{name={orange},
description={an orange coloured fruit.}}
```

and suppose you have `\gls{citrusfruit}` in your document but don't reference the `orange` entry, then the `orange` entry won't appear in your glossary until you first create the glossary and then do another run of `makeglossaries`, `makeindex` or `xindy`. For example, if the document is called `myDoc.tex`, then you must do:

```
latex myDoc
makeglossaries myDoc
latex myDoc
makeglossaries myDoc
latex myDoc
```

Likewise, an additional `makeglossaries` and L^AT_EX run may be required if the document pages shift with re-runs. For example, if the page numbering is not reset after the table of contents, the insertion of the table of contents on the second L^AT_EX run may push glossary entries across page boundaries, which means that the number lists in the glossary may need updating.

The examples in this document assume that you are accessing `makeglossaries`, `xindy` or `makeindex` via a terminal. Windows users can use the MSDOS Prompt which is usually accessed via the `Start → All Programs` menu or `Start → All Programs → Accessories` menu.

Alternatively, your text editor may have the facility to create a function that will call the required application. The article “[Glossaries, Nomenclature, List of Symbols and Acronyms](#)” in the L^AT_EX Community's⁴ Know How section describes how to do this for TeXnicCenter, and the thread “[Executing Glossaries' makeindex from a WinEdt macro](#)” on the `comp.text.tex` newsgroup describes how to do it for WinEdt. For other editors see the editor's user manual for further details.

⁴<http://www.latex-community.org/>

If any problems occur, remember to check the transcript files (e.g. `.glg` or `.alg`) for messages.

Table 3: Commands and package options that have no effect when using `xindy` or `makeindex` explicitly

Command or Package Option	<code>makeindex</code>	<code>xindy</code>
<code>order=letter</code>	use <code>-l</code>	use <code>-M ord/letorder</code>
<code>order=word</code>	default	default
<code>xindy={language=<lang>,codename=<code>}</code>	N/A	use <code>-L <lang> -C <code></code>
<code>\GlsSetXdyLanguage{<lang>}</code>	N/A	use <code>-L <lang></code>
<code>\GlsSetXdyCodePage{<code>}</code>	N/A	use <code>-C <code></code>

1.3.1 Using the `makeglossaries` Perl Script

The `makeglossaries` script picks up the relevant information from the auxiliary (`.aux`) file and will either call `xindy` or `makeindex`, depending on the supplied information. Therefore, you only need to pass the document's name without the extension to `makeglossaries`. For example, if your document is called `myDoc.tex`, type the following in your terminal:

```
latex myDoc
makeglossaries myDoc
latex myDoc
```

You may need to explicitly load `makeglossaries` into Perl:

```
perl makeglossaries myDoc
```

There is a batch file called `makeglossaries.bat` which does this for Windows users, but you must have Perl installed to be able to use it.

The `makeglossaries` script contains POD (Plain Old Documentation). If you want, you can create a man page for `makeglossaries` using `pod2man` and move the resulting file onto the man path.

1.3.2 Using `xindy` explicitly

If you want to use `xindy` to process the glossary files, you must make sure you have used the `xindy` package option:

```
\usepackage[xindy]{glossaries}
```

This is required regardless of whether you use `xindy` explicitly or whether it's called implicitly via `makeglossaries`. This causes the glossary entries to be written in raw `xindy` format, so you need to use `-I xindy` *not* `-I tex`.

To run `xindy` type the following in your terminal (all on one line):

```
xindy -L  $\langle language \rangle$  -C  $\langle encoding \rangle$  -I xindy -M  $\langle style \rangle$  -t  $\langle base \rangle$ .glg -o  
 $\langle base \rangle$ .gls  $\langle base \rangle$ .glo
```

where $\langle language \rangle$ is the required language name, $\langle encoding \rangle$ is the encoding, $\langle base \rangle$ is the name of the document without the `.tex` extension and $\langle style \rangle$ is the name of the `xindy` style file without the `.xdy` extension. The default name for this style file is $\langle base \rangle$.xdy but can be changed via `\setStyleFile{ $\langle style \rangle$ }`. You may need to specify the full path name depending on the current working directory. If any of the file names contain spaces, you must delimit them using double-quotes.

For example, if your document is called `myDoc.tex` and you are using UTF8 encoding in English, then type the following in your terminal:

```
xindy -L english -C utf8 -I xindy -M myDoc -t myDoc.glg -o myDoc.gls  
myDoc.glo
```

Note that this just creates the main glossary. You need to do the same for each of the other glossaries (including the list of acronyms if you have used the `acronym` package option), substituting `.glg`, `.gls` and `.glo` with the relevant extensions. For example, if you have used the `acronym` package option, then you would need to do:

```
xindy -L english -C utf8 -I xindy -M myDoc -t myDoc.alg -o myDoc.acr  
myDoc.acn
```

For additional glossaries, the extensions are those supplied when you created the glossary with `\newglossary`.

Note that if you use `makeglossaries` instead, you can replace all those calls to `xindy` with just one call to `makeglossaries`:

```
makeglossaries myDoc
```

Note also that some commands and package options have no effect if you use `xindy` explicitly instead of using `makeglossaries`. These are listed in [table 3](#).

1.3.3 Using `makeindex` explicitly

If you want to use `makeindex` explicitly, you must make sure that you haven't used the `xindy` package option or the glossary entries will be written in the wrong format. To run `makeindex`, type the following in your terminal:

```
makeindex -s  $\langle style \rangle$ .ist -t  $\langle base \rangle$ .glg -o  $\langle base \rangle$ .gls  $\langle base \rangle$ .glo
```

where $\langle base \rangle$ is the name of your document without the `.tex` extension and $\langle style \rangle$.ist is the name of the `makeindex` style file. By default, this is $\langle base \rangle$.ist,

but may be changed via `\setStyleFile{<style>}`. Note that there are other options, such as `-l` (letter ordering). See the `makeindex` manual for further details.

For example, if your document is called `myDoc.tex`, then type the following at the terminal:

```
makeindex -s myDoc.ist -t myDoc.glg -o myDoc.gls myDoc.glo
```

Note that this only creates the main glossary. If you have additional glossaries (for example, if you have used the `acronym` package option) then you must call `makeindex` for each glossary, substituting `.glg`, `.gls` and `.glo` with the relevant extensions. For example, if you have used the `acronym` package option, then you need to type the following in your terminal:

```
makeindex -s myDoc.ist -t myDoc.alg -o myDoc.acr myDoc.acn
```

For additional glossaries, the extensions are those supplied when you created the glossary with `\newglossary`.

Note that if you use `makeglossaries` instead, you can replace all those calls to `makeindex` with just one call to `makeglossaries`:

```
makeglossaries myDoc
```

Note also that some commands and package options have no effect if you use `makeindex` explicitly instead of using `makeglossaries`. These are listed in [table 3](#).

1.3.4 Note to Front-End and Script Developers

The information needed to determine whether to use `xindy` or `makeindex` and the information needed to call those applications is stored in the auxiliary file. This information can be gathered by a front-end, editor or script to make the glossaries where appropriate. This section describes how the information is stored in the auxiliary file.

The file extensions used by each defined glossary are given by

```
\@newglossary \@newglossary{<label>}{<log>}{<out-ext>}{<in-ext>}
```

where `<in-ext>` is the extension of the *indexing application's* input file (the output file from the `glossaries` package's point of view), `<out-ext>` is the extension of the *indexing application's* output file (the input file from the `glossaries` package's point of view) and `<log>` is the extension of the indexing application's transcript file. The label for the glossary is also given for information purposes only, but is not required by the indexing applications. For example, the information for the main glossary is written as:

```
\@newglossary{main}{glg}{gls}{glo}
```

The indexing application's style file is specified by

`\@istfilename` `\@istfilename{<filename>}`

The file extension indicates whether to use `makeindex` (`.ist`) or `xindy` (`.xdy`). Note that the glossary information is formatted differently depending on which indexing application is supposed to be used, so it's important to call the correct one.

Word or letter ordering is specified by:

`\@glsorder` `\@glsorder{<order>}`

where *<order>* can be either `word` or `letter`.

If `xindy` should be used, the language and code page for each glossary is specified by

`\@xdylanguage`
`\@gls@codepage` `\@xdylanguage{<label>}{<language>}`
`\@gls@codepage{<label>}{<code>}`

where *<label>* identifies the glossary, *<language>* is the root language (e.g. `english`) and *<code>* is the encoding (e.g. `utf8`). These commands are omitted if `makeindex` should be used.

1.4 Troubleshooting

The `glossaries` package comes with a minimal file called `minimalgls.tex` which can be used for testing. This should be located in the `samples` subdirectory (folder) of the `glossaries` documentation directory. The location varies according to your operating system and T_EX installation. For example, on my Linux partition it can be found in `/usr/local/texlive/2008/texmf-dist/doc/latex/glossaries/`. Further information on debugging L^AT_EX code is available at <http://theoval.cmp.uea.ac.uk/~nlct/latex/minexample/>.

Below is a list of the most frequently asked questions. For other queries, consult the `glossaries` FAQ at <http://theoval.cmp.uea.ac.uk/~nlct/latex/packages/faq/glossariesfaq.html>.

1. **Q.** I get the error message:

Missing `\begin{document}`

A. Check you are using an up to date version of the `xkeyval` package.

2. **Q.** I've used the `smallcaps` option, but the acronyms are displayed in normal sized upper case letters.

A. The `smallcaps` package option uses `\textsc` to typeset the acronyms. This command converts lower case letters to small capitals, while upper case

letters remain their usual size. Therefore you need to specify the acronym in lower case letters.

3. **Q.** My acronyms won't break across a line when they're expanded.

A. PDF \LaTeX can break hyperlinks across a line, but \LaTeX can't. If you can't use PDF \LaTeX then disable the first use links using the package option `hyperfirst=false`.

4. **Q.** How do I change the font that the acronyms are displayed in?

A. The easiest way to do this is to specify the `smaller` package option and redefine `\acronymfont` to use the required typesetting command. For example, suppose you would like the acronyms displayed in a sans-serif font, then you can do:

```
\usepackage[smaller]{glossaries}
\renewcommand*{\acronymfont}[1]{\textsf{#1}}
```

5. **Q.** How do I change the font that the acronyms are displayed in on first use?

A. The easiest way to do this is to specify the `smaller` package option and redefine `\firstacronymfont` to use the required command. Note that if you don't want the acronym on subsequent use to use `\textsmaller`, you will also need to redefine `\acronymfont`, as above. For example to make the acronym emphasized on first use, but use the surrounding font for subsequent use, you can do:

```
\usepackage[smaller]{glossaries}
\renewcommand*{\firstacronymfont}[1]{\emph{#1}}
\renewcommand*{\acronymfont}[1]{#1}
```

6. **Q.** I don't have Perl installed, do I have to use `makeglossaries`?

A. Although it is strongly recommended that you use `makeglossaries`, you don't have to use it. For further details, read Section 1.3.2 or Section 1.3.3, depending on whether you want to use `xindy` or `makeindex`.

7. **Q.** I'm used to using the `glossary` package: are there any instructions on migrating from the `glossary` package to the `glossaries` package?

A. Read "Upgrading from the `glossary` package to the `glossaries` package" ([glossary2glossaries.pdf](#)) which should be available from the same location as this document.

8. **Q.** I'm using `babel` but the fixed names haven't been translated.

A. The `glossaries` package currently only supports the following languages: Brazilian Portuguese, Danish, Dutch, English, French, German, Irish, Italian, Hungarian, Polish, Serbian and Spanish. If you want to add another language, send me the translations, and I'll add them to the next version.

If you are using one of the above languages, but the text hasn't been translated, try adding the `translator` package with the required languages explicitly (before you load the `glossaries` package). For example:

```
\usepackage[ngerman]{babel}
\usepackage[ngerman]{translator}
\usepackage{glossaries}
```

Alternatively, you can add the language as a global option to the class file. Check the `translator` package documentation for further details.

9. **Q.** My acronyms contain strange characters when I use commands like `\acrlong`.

A. Switch off the sanitization:

```
\usepackage[sanitize=none]{glossaries}
```

and protect fragile commands.

10. **Q.** My glossaries haven't appeared.

A. Remember to do the following:

- Add `\makeglossaries` to the document preamble.
- Use either `\printglossary` for each glossary that has been defined or `\printglossaries`.
- Use the commands listed in Section 2.4, Section 2.5 or Section 2.6 for each entry that you want to appear in the glossary.
- Run `LATEX` on your document, then run `makeglossaries`, then run `LATEX` on your document again. If you want the glossaries to appear in the table of contents, you will need an extra `LATEX` run. If any of your entries cross-reference an entry that's not referenced in the main body of the document, you will need to run `makeglossaries` (see Section 1.3) after the second `LATEX` run, followed by another `LATEX` run.

Check the log files (`.log`, `.glg` etc) for any warnings.

11. **Q.** It is possible to change the rules used to sort the glossary entries?

A. If it's for an individual entry, then you can use the entry's `sort key` to sort it according to a different term. If it's for the entire alphabet, then you will need to use `xindy` (instead of `makeindex`) and use an appropriate `xindy` language module. Writing `xindy` modules or styles is beyond the scope of this manual. Further information about `xindy` can be found at the Xindy Web Site⁵. There is also a link to the `xindy` mailing list from that site.

⁵<http://xindy.sourceforge.net/>

2 Overview of Main User Commands

This section is an overview of the main user commands and package options. If you find this too complicated, try starting out with the guide for beginners ([glossariesbegin.pdf](#)).

2.1 Package Options

The `glossaries` package options are as follows:

nowarn This suppresses all warnings generated by the `glossaries` package.

nomain This suppresses the creation of the main glossary. Note that if you use this option, you must create another glossary in which to put all your entries (either via the `acronym` package option described below or via `\newglossary` described in Section 2.9).

toc Add the glossaries to the table of contents. Note that an extra \LaTeX run is required with this option. Alternatively, you can switch this function on and off using

`\glstoctrue`

`\glstoctrue`

and

`\glstocfalse`

`\glstocfalse`

numberline When used with `toc`, this will add `\numberline{}` in the final argument of `\addcontentsline`. This will align the table of contents entry with the numbered section titles. Note that this option has no effect if the `toc` option is omitted. If `toc` is used without `numberline`, the title will be aligned with the section numbers rather than the section titles.

acronym This creates a new glossary with the label `acronym`. This is equivalent to:

```
\newglossary[alg]{acronym}{acr}{acn}{\acronymname}
```

If the `acronym` package option is used, `\acronymtype` is set to `acronym` otherwise it is set to `main`.⁶ Entries that are defined using `\newacronym` are placed in the glossary whose label is given by `\acronymtype`, unless another glossary is explicitly specified.

⁶Actually it sets `\acronymtype` to `\glsdefaulttype` if the `acronym` package option is not used, but `\glsdefaulttype` usually has the value `main`.

acronymlists By default, only the `acronym` glossary is considered to be a list of acronyms. If you have other lists of acronyms, you can specify them as a comma-separated list in the value of `acronymlists`. For example, if you want the `main` glossary to also contain a list of acronyms, you can do:

```
\usepackage[acronym,acronymlists={main}]{glossaries}
```

No check is performed to determine if the listed glossaries exist, so you can add glossaries you haven't defined yet. For example:

```
\usepackage[acronym,acronymlists={main,acronym2}]{glossaries}
\newglossary[alg2]{acronym2}{acr2}{acn2}{Statistical Acronyms}
```

section This is a $\langle key \rangle = \langle value \rangle$ option. Its value should be the name of a sectional unit (e.g. chapter). This will make the glossaries appear in the named sectional unit, otherwise each glossary will appear in a chapter, if chapters exist, otherwise in a section. Unnumbered sectional units will be used by default. Example:

```
\usepackage[section=subsection]{glossaries}
```

You can omit the value if you want to use sections, i.e.

```
\usepackage[section]{glossaries}
```

is equivalent to

```
\usepackage[section=section]{glossaries}
```

You can change this value later in the document using

`\setglossarysection`

```
\setglossarysection{\langle name \rangle}
```

where $\langle name \rangle$ is the sectional unit.

The start of each glossary adds information to the page header via

`\glossarymark`

```
\glossarymark{\langle glossary title \rangle}
```

This defaults to `\@mkboth`, but you may need to redefine it. For example, to only change the right header:

```
\renewcommand{\glossarymark}[1]{\markright{#1}}
```

or to prevent it from changing the headers:

```
\renewcommand{\glossarymark}[1]{}
```


Occasionally you may find that another package defines `\cleardoublepage` when it is not required. This may cause an unwanted blank page to appear before each glossary. This can be fixed by redefining `\glsclearpage`:

```
\renewcommand*{\glsclearpage}{\clearpage}
```

numberedsection The glossaries are placed in unnumbered sectional units by default, but this can be changed using `numberedsection`. This option can take three possible values: `false` (no number, i.e. use starred form), `nolabel` (numbered, i.e. unstarred form, but not labelled) and `autolabel` (numbered with automatic labelling). If `numberedsection=autolabel` is used, each glossary is given a label that matches the glossary type, so the main (default) glossary is labelled `main`, the list of acronyms is labelled `acronym`⁷ and additional glossaries are labelled using the value specified in the first mandatory argument to `\newglossary`. For example, if you load `glossaries` using:

```
\usepackage[section,numberedsection=autolabel]{glossaries}
```

then each glossary will appear in a numbered section, and can be referenced using something like:

The main glossary is in section~\ref{main} and the list of acronyms is in section~\ref{acronym}.

If you can't decide whether to have the acronyms in the main glossary or a separate list of acronyms, you can use `\acronymtype` which is set to `main` if the `acronym` option is not used and is set to `acronym` if the `acronym` option is used. For example:

The list of acronyms is in section~\ref{\acronymtype}.

As from version 1.14, you can add a prefix to the label by redefining

`\glsautoprefix`

```
\glsautoprefix
```

For example:

```
\renewcommand*{\glsautoprefix}{glo:}
```

will add `glo:` to the automatically generated label, so you can then, for example, refer to the list of acronyms as follows:

The list of acronyms is in section~\ref{glo:\acronymtype}.

⁷if the `acronym` option is used, otherwise the list of acronyms is the main glossary

Or, if you are undecided on a prefix:

The list of acronyms is in section~\ref{\glsautoprefix\acronymtype}.

style This is a $\langle key \rangle = \langle value \rangle$ option. Its value should be the name of the glossary style to use. Predefined glossary styles are listed in Section 2.12.

nolong This prevents the `glossaries` package from automatically loading `glossary-long` (which means that the `longtable` package also won't be loaded). This reduces overhead by not defining unwanted styles and commands. Note that if you use this option, you won't be able to use any of the glossary styles defined in the `glossary-long` package.

nosuper This prevents the `glossaries` package from automatically loading `glossary-super` (which means that the `supertabular` package also won't be loaded). This reduces overhead by not defining unwanted styles and commands. Note that if you use this option, you won't be able to use any of the glossary styles defined in the `glossary-super` package.

nolist This prevents the `glossaries` package from automatically loading `glossary-list`. This reduces overhead by not defining unwanted styles. Note that if you use this option, you won't be able to use any of the glossary styles defined in the `glossary-list` package. Note that since the default style is `list`, you will also need to use the `style` option to set the style to something else.

notree This prevents the `glossaries` package from automatically loading `glossary-tree`. This reduces overhead by not defining unwanted styles. Note that if you use this option, you won't be able to use any of the glossary styles defined in the `glossary-tree` package.

nostyles This prevents all the predefined styles from being loaded. This option is provided in the event that the user has custom styles that are not dependent on the styles provided by the `glossaries` package. Note that if you use this option, you can't use the `style` package option. Instead you must either use `\glossarystyle{\langle style \rangle}` or the `style` key in the optional argument to `\printglossary`.

nonumberlist This option will suppress the associated number lists in the glossaries (see also Section 2.3).

counter This is a $\langle key \rangle = \langle value \rangle$ option. The value should be the name of the default counter to use in the number lists.

sanitize This is a $\langle key \rangle = \langle value \rangle$ option whose value is also a $\langle key \rangle = \langle value \rangle$ list. By default, the `glossaries` package sanitizes the values of the `name`, `description` and `symbol` keys used when defining a new glossary entry. This means that you can use fragile commands in those keys, but it may lead to unexpected results if you try to display these values within the document text. This sanitization can be switched off using the `sanitize` package option. (See Section 4.2 and

Section 4.7 for further details.) For example, to switch off the sanitization for the description and name keys, but not for the symbol key, do:

```
\usepackage[sanitize={name=false,description=false,%  
symbol=true}]{glossaries}
```

You can use `sanitize=none` as a shortcut for `sanitize={name=false,description=false,symbol=false}`.

Note: this sanitization only applies to the name, description and symbol keys. It doesn't apply to any of the other keys (except the `sort` key which is always sanitized) so fragile commands contained in the value of the other keys must always be protected using `\protect`. Since the value of the `text` key is obtained from the `name` key, you will still need to protect fragile commands in the `name` key if you don't use the `text` key.

description This option changes the definition of `\newacronym` to allow a description. See Section 2.10 for further details.

footnote This option changes the definition of `\newacronym` and the way that acronyms are displayed. See Section 2.10 for further details.

smallcaps This option changes the definition of `\newacronym` and the way that acronyms are displayed. See Section 2.10 for further details.

smaller This option changes the definition of `\newacronym` and the way that acronyms are displayed. If you use this option, you will need to include the `relsize` package or otherwise define `\textsmaller` or redefine `\acronymfont`. See Section 2.10 for further details.

dua This option changes the definition of `\newacronym` so that acronyms are always expanded. See Section 2.10 for further details.

shortcuts This option provides shortcut commands for acronyms. See Section 2.10 for further details.

makeindex (Default) The glossary information and indexing style file will be written in `makeindex` format. If you use `makeglossaries`, it will automatically detect that it needs to call `makeindex`. If you don't use `makeglossaries`, you need to remember to use `makeindex` not `xindy`. The indexing style file will be given a `.ist` extension.

xindy The glossary information and indexing style file will be written in `xindy` format. If you use `makeglossaries`, it will automatically detect that it needs to call `xindy`. If you don't use `makeglossaries`, you need to remember to use `xindy` not `makeindex`. The indexing style file will be given a `.xdy` extension.

The xindy package option may additionally have a value that is a $\langle key \rangle = \langle value \rangle$ comma-separated list to override the language and codepage. For example:

```
\usepackage[xindy={language=english,codepage=utf8}]{glossaries}
```

You can also specify whether you want a number group in the glossary. This defaults to true, but can be suppressed. For example:

```
\usepackage[xindy={glsnumbers=false}]{glossaries}
```

See Section 2.8.2 for further details on using xindy with the glossaries package.

order This may take two values: `word` or `letter`. The default is word ordering. Note that this option has no effect if you don't use `makeglossaries`.

translate This is a boolean option. The default is true if `babel`, `polyglossia` or `translator` have been loaded, otherwise the default value is false.

translate=true If `babel` has been loaded and the `translator` package is installed, `translator` will be loaded and the translations will be provided by the `translator` package interface. You can modify the translations by providing your own dictionary. If the `translator` package isn't installed and `babel` is loaded, the `glossaries-babel` package will be loaded and the translations will be provided using `babel`'s `\addto\caption{language}` mechanism. If `polyglossia` has been loaded, `glossaries-polyglossia` will be loaded.

translate=false Don't provide translations, even if `babel` or `polyglossia` has been loaded. You can then provide your own translations or explicitly load `glossaries-babel` or `glossaries-polyglossia`.

See Section 1.2.1 for further details.

hyperfirst This is a boolean option that specifies whether each term has a hyperlink on first use. The default is `hyperfirst=true` (terms on first use have a hyperlink, unless explicitly suppressed using starred versions of commands such as `\gls*`).

2.2 Defining Glossary Entries

All glossary entries must be defined before they are used, so it is better to define them in the preamble to ensure this.⁸ However only those entries that occur in the document (using any of the commands described in Section 2.4, Section 2.5 or Section 2.6) will appear in the glossary. Each time an entry is used in this way, a line is added to an associated glossary file (`.glo`), which then needs to be converted

⁸The only preamble restriction on `\newglossaryentry` and `\newacronym` was removed in version 1.13, but the restriction remains for `\loadglsentries`.

into a corresponding `.gls` file which contains the typeset glossary which is input by `\printglossary` or `\printglossaries`. The Perl script `makeglossaries` can be used to call `makeindex` or `xindy`, using a customised indexing style file, for each of the glossaries that are defined in the document. Note that there should be no need for you to explicitly edit or input any of these external files. See Section 1.3 for further details.

The command

`\makeglossaries`

```
\makeglossaries
```

must be placed in the preamble in order to create the customised `makeindex` (`.ist`) or `xindy` (`.xdy`) style file and to ensure that glossary entries are written to the appropriate output files. If you omit `\makeglossaries` none of the glossaries will be created.

Note that some of the commands provided by the `glossaries` package must be placed before `\makeglossaries` as they are required when creating the customised style file. If you attempt to use those commands after `\makeglossaries` you will generate an error.

You can suppress the creation of the customised `xindy` or `makeindex` style file using

`\noist`

```
\noist
```

Note that this command must be used before `\makeglossaries`.

The default name for the customised style file is given by `\jobname.ist` (for `makeindex`) or `\jobname.xdy` (for `xindy`). This name may be changed using:

`\setStyleFile`

```
\setStyleFile{<name>}
```

where `<name>` is the name of the style file without the extension. Note that this command must be used before `\makeglossaries`.

Each glossary entry is assigned a number list that lists all the locations in the document where that entry was used. By default, the location refers to the page number but this may be overridden using the `counter` package option. The default form of the location number assumes a full stop compositor (e.g. 1.2), but if your location numbers use a different compositor (e.g. 1-2) you need to set this using

`\glsSetCompositor`

```
\glsSetCompositor{<symbol>}
```

For example:

```
\glsSetCompositor{-}
```

Note that this command must be used before `\makeglossaries`.

If you use `xindy`, you can have a different compositor for page numbers starting with an uppercase alphabetical character using:

`\glsSetAlphaCompositor`

```
\glsSetAlphaCompositor{<symbol>}
```

Note that this command has no effect if you haven't used the `xindy` package option. For example, if you want number lists containing a mixture of A-1 and 2.3 style formats, then do:

```
\glsSetCompositor{.} \glsSetAlphaCompositor{-}
```

See Section 2.3 for further information about number lists.

New glossary entries are defined using the command:

`\newglossaryentry`

```
\newglossaryentry{<label>}{<key-val list>}
```

The first argument, `<label>`, must be a unique label with which to identify this entry. The second argument, `<key-val list>`, is a `<key>=<value>` list that supplies the relevant information about this entry. There are two required fields: `description` and either `name` or `parent`. Available fields are listed below:

name The name of the entry (as it will appear in the glossary). If this key is omitted and the `parent` key is supplied, this value will be the same as the parent's name.

description A brief description of this term (to appear in the glossary). Within this value, you can use

`\nopostdesc`

```
\nopostdesc
```

to suppress the description terminator for this entry. For example, if this entry is a parent entry that doesn't require a description, you can do `description={\nopostdesc}`. If you want a paragraph break in the description use

`\glspar`

```
\glspar
```

However, note that not all glossary styles support multi-line descriptions. If you are using one of the tabular-like glossary styles that permit multi-line descriptions, use `\newline` not `\\` if you want to force a line break.

parent The label of the parent entry. Note that the parent entry must be defined before its sub-entries. See Section 2.2.2 for further details.

- descriptionplural** The plural form of the description (as passed to `\glsdisplay` and `\glsdisplayfirst` by `\glspl`, `\Glspl` and `\GLSp1`). If omitted, the value is set to the same as the **description** key.
- text** How this entry will appear in the document text when using `\gls` (or one of its uppercase variants). If this field is omitted, the value of the **name** key is used.
- first** How the entry will appear in the document text the first time it is used with `\gls` (or one of its uppercase variants). If this field is omitted, the value of the **text** key is used.
- plural** How the entry will appear in the document text when using `\glspl` (or one of its uppercase variants). If this field is omitted, the value is obtained by appending `\glspluralsuffix` to the value of the **text** field. The default value of `\glspluralsuffix` is the letter “s”.
- firstplural** How the entry will appear in the document text the first time it is used with `\glspl` (or one of its uppercase variants). If this field is omitted, the value is obtained from the **plural** key, if the **first** key is omitted, or by appending `\glspluralsuffix` to the value of the **first** field, if the **first** field is present.
- Note:** prior to version 1.13, the default value of **firstplural** was always taken by appending “s” to the **first** key, which meant that you had to specify both **plural** and **firstplural**, even if you hadn’t used the **first** key.
- symbol** This field is provided to allow the user to specify an associated symbol. If omitted, the value is set to `\relax`. Note that not all glossary styles display the symbol.
- symbolplural** This is the plural form of the symbol (as passed to `\glsdisplay` and `\glsdisplayfirst` by `\glspl`, `\Glspl` and `\GLSp1`). If omitted, the value is set to the same as the **symbol** key.
- sort** This value indicates how `makeindex` or `xindy` should sort this entry. If omitted, the value is given by the **name** field.
- type** This specifies the label of the glossary in which this entry belongs. If omitted, the default glossary is assumed. The list of acronyms type is given by `\acronymtype` which will either be **main** or **acronym**, depending on whether the **acronym** package option was used.
- user1, ..., user6** Six additional keys provided for any additional information the user may want to specify. (For example, an associated dimension or an alternative plural.)
- nonumberlist** Suppress the number list for this entry.

see Cross-reference another entry. Using the `see` key will automatically add this entry to the glossary, but will not automatically add the cross-referenced entry. The referenced entry should be supplied as the value to this key. If you want to override the “see” tag, you can supply the new tag in square brackets before the label. For example `see=[see also]{anotherlabel}`. For further details, see Section 2.6.

Note that if the name starts with an accented letter or non-Latin character, you must group the accented letter, otherwise it will cause a problem for commands like `\Gls` and `\Glspl`. For example:

```
\newglossaryentry{elite}{name={{\'}elite},
description={select group or class}}
```

Note that the same applies if you are using the `inputenc` package:

```
\newglossaryentry{elite}{name={{\'}elite},
description={select group or class}}
```

Note that in both of the above examples, you will also need to supply the `sort` key if you are using `makeindex` whereas `xindy` is usually able to sort accented letters correctly.

2.2.1 Plurals

You may have noticed from above that you can specify the plural form when you define a term. If you omit this, the plural will be obtained by appending

`\glspluralsuffix`

`\glspluralsuffix`

to the singular form. This command defaults to the letter “s”. For example:

```
\newglossaryentry{cow}{name=cow,description={a fully grown
female of any bovine animal}}
```

defines a new entry whose singular form is “cow” and plural form is “cows”. However, if you are writing in archaic English, you may want to use “kine” as the plural form, in which case you would have to do:

```
\newglossaryentry{cow}{name=cow,plural=kine,
description={a fully grown female of any bovine animal}}
```

If you are writing in a language that supports multiple plurals (for a given term) then use the `plural` key for one of them and one of the user keys to specify the other plural form. For example:

```
\newglossaryentry{cow}{name=cow,description={a fully grown
female of any bovine animal (plural cows, archaic plural kine)},
user1={kine}}
```


You can then use `\glspl{cow}` to produce “cows” and `\glsuseri{cow}` to produce “kine”. You can, of course, define an easy to remember synonym. For example:

```
\let\glsaltpl\glsuseri
```

Then you don’t have to remember which key you used to store the alternative plural.

If you are using a language that usually forms plurals by appending a different letter, or sequence of letters, you can redefine `\glspluralsuffix` as required. However, this must be done *before* the entries are defined. For languages that don’t form plurals by simply appending a suffix, all the plural forms must be specified using the `plural` key (and the `firstplural` key where necessary).

2.2.2 Sub-Entries

As from version 1.17, it is possible to specify sub-entries. These may be used to order the glossary into categories, in which case the sub-entry will have a different name to its parent entry, or it may be used to distinguish different definitions for the same word, in which case the sub-entries will have the same name as the parent entry. Note that not all glossary styles support hierarchical entries and may display all the entries in a flat format. Of the styles that support sub-entries, some display the sub-entry’s name whilst others don’t. Therefore you need to ensure that you use a suitable style. See Section 2.12 for a list of predefined styles.

Note that the parent entry will automatically be added to the glossary if any of its child entries are used in the document. If the parent entry is not referenced in the document, it will not have a number list.

Hierarchical Categories To arrange a glossary with hierarchical categories, you need to first define the category and then define the sub-entries using the relevant category entry as the value of the `parent` key. For example, suppose I want a glossary of mathematical symbols that are divided into Greek letters and Roman letters. Then I can define the categories as follows:

```
\newglossaryentry{greekletter}{name={Greek letters},
description={\nopostdesc}}
```

```
\newglossaryentry{romanletter}{name={Roman letters},
description={\nopostdesc}}
```

Note that in this example, the category entries don’t need a description so I have set the descriptions to `\nopostdesc`. This gives a blank description and suppresses the description terminator.

I can now define my sub-entries as follows:

```
\newglossaryentry{pi}{name={ $\pi$ },
description={ratio of the circumference of a circle to the diameter},
parent=greekletter}
```

```
\newglossaryentry{C}{name=C,
```

```
description={Euler's constant},
parent=romanletter}
```

Homographs Sub-entries that have the same name as the parent entry, don't need to have the `name` key. For example, the word “glossary” can mean a list of technical words or a collection of glosses. In both cases the plural is “glossaries”. So first define the parent entry:

```
\newglossaryentry{glossary}{name=glossary,
description={\nopostdesc},
plural={glossaries}}
```

Again, the parent entry has no description, so the description terminator needs to be suppressed using `\nopostdesc`.

Now define the two different meanings of the word:

```
\newglossaryentry{glossarylist}{
description={1) list of technical words},
sort={1},
parent={glossary}}
```

```
\newglossaryentry{glossarycol}{
description={2) collection of glosses},
sort={2},
parent={glossary}}
```

Note that if I reference the parent entry, the location will be added to the parent's number list, whereas if I reference any of the child entries, the location will be added to the child entry's number list. Note also that since the sub-entries have the same name, the `sort` key is required.

In the above example, the plural form for both of the child entries is the same as the parent entry, so the `plural` key was not required for the child entries. However, if the sub-entries have different plurals, they will need to be specified. For example:

```
\newglossaryentry{bravo}{name={bravo},
description={\nopostdesc}}
```

```
\newglossaryentry{bravocry}{description={1) cry of approval (pl.\ bravos)},
sort={1},
plural={bravos},
parent=bravo}
```

```
\newglossaryentry{bravoruffian}{description={2) hired ruffian or
killer (pl.\ bravoos)},
sort={2},
plural={bravoos},
parent=bravo}
```

2.2.3 Loading Entries From a File

You can store all your glossary entry definitions in another file and use:

```
\loadglsentries \loadglsentries[<type>]{<filename>}
```

where *<filename>* is the name of the file containing all the `\newglossaryentry` commands. The optional argument *<type>* is the name of the glossary to which those entries should belong, for those entries where the `type` key has been omitted (or, more specifically, for those entries whose type has been specified by `\glsdefaulttype`, which is what `\newglossaryentry` uses by default). For example, suppose I have a file called `myentries.tex` which contains:

```
\newglossaryentry{perl}{type=main,
name={Perl},
description={A scripting language}}

\newglossaryentry{tex}{name={\TeX},
description={A typesetting language},sort={TeX}}

\newglossaryentry{html}{type=\glsdefaulttype,
name={html},
description={A mark up language}}
```

and suppose in my document preamble I use the command:

```
\loadglsentries[languages]{myentries}
```

then this will add the entries `tex` and `html` to the glossary whose type is given by `languages`, but the entry `perl` will be added to the main glossary, since it explicitly sets the type to `main`.

Note: if you use `\newacronym` (see Section 2.10) the type is set as `type=\acronymtype` unless you explicitly override it. For example, if my file `myacronyms.tex` contains:

```
\newacronym{aca}{aca}{a contrived acronym}
```

then (supposing I have defined a new glossary type called `altacronym`)

```
\loadglsentries[altacronym]{myacronyms}
```

will add `aca` to the glossary type `acronym`, if the package option `acronym` has been specified, or will add `aca` to the glossary type `altacronym`, if the package option `acronym` is not specified.⁹ In this instance, it is better to change `myacronyms.tex` to:

```
\newacronym[type=\glsdefaulttype]{aca}{aca}{a contrived acronym}
```

and now

```
\loadglsentries[altacronym]{myacronyms}
```

⁹This is because `\acronymtype` is set to `\glsdefaulttype` if the `acronym` package option is not used.

will add `aca` to the glossary type `altacronym`, regardless of whether or not the package option `acronym` is used.

Note that only those entries that have been used in the text will appear in the relevant glossaries. Note also that `\loadglsentries` may only be used in the preamble.

2.3 Number lists

Each entry in the glossary has an associated *number list*. By default, these numbers refer to the pages on which that entry has been used (using any of the commands described in Section 2.4 and Section 2.5). The number list can be suppressed using the `nonumberlist` package option, or an alternative counter can be set as the default using the `counter` package option. The number list is also referred to as the location list.

Both `makeindex` and `xindy` concatenate a sequence of 3 or more consecutive pages into a range. With `xindy` you can vary the minimum sequence length using `\GlsSetXdyMinRangeLength{<n>}` where `<n>` is either an integer or the keyword `none` which indicates that there should be no range formation.

Note that `\GlsSetXdyMinRangeLength` must be used before `\makeglossaries` and has no effect if `\noist` is used.

With both `makeindex` and `xindy`, you can replace the separator and the closing number in the range using:

`\glsSetSuffixF`

`\glsSetSuffixF{<suffix>}`

`\glsSetSuffixFF`

`\glsSetSuffixFF{<suffix>}`

where the former command specifies the suffix to use for a 2 page list and the latter specifies the suffix to use for longer lists. For example:

```
\glsSetSuffixF{f.}
\glsSetSuffixFF{ff.}
```

Note that if you use `xindy`, you will also need to set the minimum range length to 1 if you want to change these suffixes:

```
\GlsSetXdyMinRangeLength{1}
```

Note that if you use the `hyperref` package, you will need to use `\nohyperpage` in the suffix to ensure that the hyperlinks work correctly. For example:

```
\glsSetSuffixF{\nohyperpage{f.}}
\glsSetSuffixFF{\nohyperpage{ff.}}
```

Note that `\glsSetSuffixF` and `\glsSetSuffixFF` must be used before `\makeglossaries` and have no effect if `\noist` is used.

2.4 Links to Glossary Entries

Once you have defined a glossary entry using `\newglossaryentry`, you can refer to that entry in the document using one of the commands listed in this section. The text which appears at that point in the document when using one of these commands is referred to as the *link text* (even if there are no hyperlinks). The commands in this section also add a line to an external file that is used by `makeindex` or `xindy` to generate the relevant entry in the glossary. This information includes an associated location that is added to the number list for that entry. By default, the location refers to the page number. For further information on number lists, see Section 2.3.

It is strongly recommended that you don't use the commands defined in this section in the arguments of sectioning or caption commands.

The above warning is particularly important if you are using the `glossaries` package in conjunction with the `hyperref` package. Instead, use one of the commands listed in Section 2.7 (such as `\glsentrytext`) or provide an alternative via the optional argument to the sectioning/caption command. Examples:

```
\section{An overview of \glsentrytext{perl}}
\section{An overview of Perl}{An overview of \gls{perl}}
```

The way the link text is displayed depends on

`\glstextformat`

`\glstextformat{<text>}`

For example, to make all link text appear in a sans-serif font, do:

```
\renewcommand*{\glstextformat}[1]{\textsf{#1}}
```

Each entry has an associated conditional referred to as the first use flag. This determines whether `\gls`, `\glsp` (and their uppercase variants) should use the value of the `first` or `text` keys. Note that an entry can be used without affecting the first use flag (for example, when used with `\glslink`). See Section 2.11 for commands that unset or reset this conditional.

The command:

`\glslink`

`\glslink[<options>]{<label>}{<text>}`

will place `\glstextformat{<text>}` in the document at that point and add a line into the associated glossary file for the glossary entry given by `<label>`. If hyperlinks

are supported, $\langle text \rangle$ will be a hyperlink to the relevant line in the glossary. (Note that this command doesn't affect the first use flag: use `\glsdisp` instead.) The optional argument $\langle options \rangle$ must be a $\langle key \rangle = \langle value \rangle$ list which can take any of the following keys:

format This specifies how to format the associated location number for this entry in the glossary. This value is equivalent to the `makeindex` `encap` value, and (as with `\index`) the value needs to be the name of a command *without* the initial backslash. As with `\index`, the characters `(` and `)` can also be used to specify the beginning and ending of a number range. Again as with `\index`, the command should be the name of a command which takes an argument (which will be the associated location). Be careful not to use a declaration (such as `\bfseries`) instead of a text block command (such as `\textbf`) as the effect is not guaranteed to be localised. If you want to apply more than one style to a given entry (e.g. **bold** and *italic*) you will need to create a command that applies both formats, e.g.

```
\newcommand*{\textbfem}[1]{\textbf{\emph{#1}}}
```

and use that command.

In this document, the standard formats refer to the standard text block commands such as `\textbf` or `\emph` or any of the commands listed in [table 4](#).

If you use `xindy` instead of `makeindex`, you must specify any non-standard formats that you want to use with the `format` key using `\GlsAddXdyAttribute{<name>}`. So if you use `xindy` with the above example, you would need to add:

```
\GlsAddXdyAttribute{textbfem}
```

Note that unlike `\index`, you can't have anything following the command name, such as an asterisk or arguments. If you want to cross-reference another entry, either use the `see` key when you define the entry or use `\glssee` (described in [Section 2.6](#)).

If you are using hyperlinks and you want to change the font of the hyperlinked location, don't use `\hyperpage` (provided by the `hyperref` package) as the locations may not refer to a page number. Instead, the `glossaries` package provides number formats listed in [table 4](#).

Note that if the `\hyperlink` command hasn't been defined, the `\hyper{<xx>}` formats are equivalent to the analogous `\text{<xx>}` font commands (and `\hyperemph` is equivalent to `\emph`). If you want to make a new format, you will need to define a command which takes one argument and use that. For

Table 4: Predefined Hyperlinked Location Formats

<code>hyperarm</code>	serif hyperlink
<code>hypersf</code>	sans-serif hyperlink
<code>hypertt</code>	monospaced hyperlink
<code>hyperbf</code>	bold hyperlink
<code>hypermd</code>	medium weight hyperlink
<code>hyperit</code>	italic hyperlink
<code>hypersl</code>	slanted hyperlink
<code>hyperup</code>	upright hyperlink
<code>hypersc</code>	small caps hyperlink
<code>hyperemph</code>	emphasized hyperlink

example, if you want the location number to be in a bold sans-serif font, you can define a command called, say, `\hyperbsf`:

```
\newcommand{\hyperbsf}[1]{\textbf{\hypersf{#1}}}
```

and then use `hyperbsf` as the value for the `format` key. (See also Section 4.15.) Remember that if you use `xindy`, you will need to add this to the list of location attributes:

```
\GlsAddXdyAttribute{hyperbsf}
```

counter This specifies which counter to use for this location. This overrides the default counter used by this entry. (See also Section 2.3.)

hyper This is a boolean key which can be used to enable/disable the hyperlink to the relevant entry in the glossary. (Note that setting `hyper=true` will have no effect if `\hyperlink` has not been defined.) The default value is `hyper=true`.

There is also a starred version:

```
\glslink*[\<options>]{\<label>}{\<text>}
```

which is equivalent to `\glslink`, except it sets `hyper=false`. Similarly, all the following commands described in this section also have a starred version that disables the hyperlink.

The command:

```
\gls[\<options>]{\<label>}{\<insert>}
```

is the same as `\glslink`, except that the link text is determined from the values of the `text` and `first` keys supplied when the entry was defined using

`\newglossaryentry`. If the entry has been marked as having been used, the value of the `text` key will be used, otherwise the value of the `first` key will be used. On completion, `\gls` will mark the entry's first use flag as used.

There are two uppercase variants:

`\Gls` `\Gls[<options>]{<label>}[<insert>]`

and

`\GLS` `\GLS[<options>]{<label>}[<insert>]`

which make the first letter of the link text or all the link text uppercase, respectively.

The final optional argument *<insert>*, allows you to insert some additional text into the link text. By default, this will append *<insert>* at the end of the link text, but this can be changed (see Section 2.4.1).

The first optional argument *<options>* is the same as the optional argument to `\glslink`. As with `\glslink`, these commands also have a starred version that disable the hyperlink.

There are also analogous plural forms:

`\glspl` `\glspl[<options>]{<label>}[<insert>]`

`\Glspl` `\Glspl[<options>]{<label>}[<insert>]`

`\GLSpl` `\GLSpl[<options>]{<label>}[<insert>]`

These determine the link text from the `plural` and `firstplural` keys supplied when the entry was first defined. As before, these commands also have a starred version that disable the hyperlink.

Note that `\glslink` doesn't use or affect the first use flag, nor does it use `\glsdisplay` or `\glsdisplayfirst` (see Section 2.4.1). Instead, you can use:

`\glsdisp` `\glsdisp[<options>]{<label>}{<link text>}`

This behaves in the same way as `\gls`, except that it uses *<link text>* instead of the value of the `first` or `text` key. (Note that this command always sets *<insert>* to nothing.) This command affects the first use flag, and uses `\glsdisplay` or `\glsdisplayfirst`.

The command:

`\glstext` `\glstext[<options>]{<label>}[<insert>]`

is similar to `\gls` except that it always uses the value of the `text` key and does not affect the first use flag. Unlike `\gls`, the inserted text *<insert>* is always appended to the link text since `\glstext` doesn't use `\glsdisplay` or `\glsdisplayfirst`. (The same is true for all the following commands described in this section.)

There are also analogous commands:

`\Glstext` `\Glstext[<options>]{<text>}[<insert>]`

`\GLStext` `\GLStext[<options>]{<text>}[<insert>]`

As before, these commands also have a starred version that disable the hyperlink.

The command:

`\glsfirst` `\glsfirst[<options>]{<label>}[<insert>]`

is similar to `\glstext` except that it always uses the value of the `first` key. Again, *<insert>* is always appended to the end of the link text and does not affect the first use flag.

There are also analogous commands:

`\Glsfirst` `\Glsfirst[<options>]{<text>}[<insert>]`

`\GLSfirst` `\GLSfirst[<options>]{<text>}[<insert>]`

As before, these commands also have a starred version that disable the hyperlink.

The command:

`\glsplural` `\glsplural[<options>]{<label>}[<insert>]`

is similar to `\glstext` except that it always uses the value of the `plural` key. Again, *<insert>* is always appended to the end of the link text and does not mark the entry as having been used.

There are also analogous commands:

`\Glsplural` `\Glsplural[<options>]{<text>}[<insert>]`

`\GLSplural` `\GLSplural[<options>]{<text>}[<insert>]`

As before, these commands also have a starred version that disable the hyperlink.
The command:

`\glsfirstplural` `\glsfirstplural[<options>]{<label>}[<insert>]`

is similar to `\glstext` except that it always uses the value of the `firstplural` key. Again, *<insert>* is always appended to the end of the link text and does not mark the entry as having been used.

There are also analogous commands:

`\Glsfirstplural` `\Glsfirstplural[<options>]{<text>}[<insert>]`

`\GLSfirstplural` `\GLSfirstplural[<options>]{<text>}[<insert>]`

As before, these commands also have a starred version that disable the hyperlink.
The command:

`\glsname` `\glsname[<options>]{<label>}[<insert>]`

is similar to `\glstext` except that it always uses the value of the `name` key. Again, *<insert>* is always appended to the end of the link text and does not mark the entry as having been used. Note: if you want to use this command and the `name` key contains commands, you will have to disable the **sanitization** of the `name` key and protect fragile commands.

There are also analogous commands:

`\Glsname` `\Glsname[<options>]{<text>}[<insert>]`

`\GLSname` `\GLSname[<options>]{<text>}[<insert>]`

As before, these commands also have a starred version that disable the hyperlink.
The command:

`\glssymbol` `\glssymbol[<options>]{<label>}[<insert>]`

is similar to `\glstext` except that it always uses the value of the `symbol` key. Again, *<insert>* is always appended to the end of the link text and does not mark the entry as having been used. Note: if you want to use this command and the `symbol` key contains commands, you will have to disable the **sanitization** of the `symbol` key and protect fragile commands.

There are also analogous commands:

`\Glssymbol` `\Glssymbol[<options>]{<text>}[<insert>]`

`\GLSsymbol` `\GLSsymbol[<options>]{<text>}[<insert>]`

As before, these commands also have a starred version that disable the hyperlink.
The command:

`\glldesc` `\glldesc[<options>]{<label>}[<insert>]`

is similar to `\glstext` except that it always uses the value of the `description` key. Again, *<insert>* is always appended to the end of the link text and does not mark the entry as having been used. Note: if you want to use this command and the `description` key contains commands, you will have to disable the **sanitization** of the `description` key and protect fragile commands.

There are also analogous commands:

`\Glsdesc` `\Glsdesc[<options>]{<text>}[<insert>]`

`\GLSdesc` `\GLSdesc[<options>]{<text>}[<insert>]`

As before, these commands also have a starred version that disable the hyperlink.
The command:

`\gluseri` `\gluseri[<options>]{<label>}[<insert>]`

is similar to `\glstext` except that it always uses the value of the `user1` key. Again, *<insert>* is always appended to the end of the link text and does not mark the entry as having been used.

There are also analogous commands:

`\Glsuseri` `\Glsuseri[<options>]{<text>}[<insert>]`

`\GLSuseri` `\GLSuseri[<options>]{<text>}[<insert>]`

As before, these commands also have a starred version that disable the hyperlink.
Similarly for the other user keys:

`\gluserii` `\gluserii[<options>]{<text>}[<insert>]`

<code>\Glsuserii</code>	<code>\Glsuserii[<i><options></i>]{<i><text></i>}[<i><insert></i>]</code>
<code>\GLSuserii</code>	<code>\GLSuserii[<i><options></i>]{<i><text></i>}[<i><insert></i>]</code>
<code>\glsuseriii</code>	<code>\glsuseriii[<i><options></i>]{<i><text></i>}[<i><insert></i>]</code>
<code>\Glsuseriii</code>	<code>\Glsuseriii[<i><options></i>]{<i><text></i>}[<i><insert></i>]</code>
<code>\GLSuseriii</code>	<code>\GLSuseriii[<i><options></i>]{<i><text></i>}[<i><insert></i>]</code>
<code>\glsuseriv</code>	<code>\glsuseriv[<i><options></i>]{<i><text></i>}[<i><insert></i>]</code>
<code>\Glsuseriv</code>	<code>\Glsuseriv[<i><options></i>]{<i><text></i>}[<i><insert></i>]</code>
<code>\GLSuseriv</code>	<code>\GLSuseriv[<i><options></i>]{<i><text></i>}[<i><insert></i>]</code>
<code>\glsuserv</code>	<code>\glsuserv[<i><options></i>]{<i><text></i>}[<i><insert></i>]</code>
<code>\Glsuserv</code>	<code>\Glsuserv[<i><options></i>]{<i><text></i>}[<i><insert></i>]</code>
<code>\GLSuserv</code>	<code>\GLSuserv[<i><options></i>]{<i><text></i>}[<i><insert></i>]</code>
<code>\glsuservi</code>	<code>\glsuservi[<i><options></i>]{<i><text></i>}[<i><insert></i>]</code>
<code>\Glsuservi</code>	<code>\Glsuservi[<i><options></i>]{<i><text></i>}[<i><insert></i>]</code>

`\GLSuservi` `\GLSuservi[\langle options \rangle]{\langle text \rangle}[\langle insert \rangle]`

2.4.1 Changing the format of the link text

The format of the link text for `\gls`, `\glspl` and their uppercase variants is governed by two commands:

`\glsdisplayfirst` `\glsdisplayfirst{\langle first/first plural \rangle}{\langle description \rangle}{\langle symbol \rangle}{\langle insert \rangle}`

which is used the first time a glossary entry is used in the text and

`\glsdisplay` `\glsdisplay{\langle text/plural \rangle}{\langle description \rangle}{\langle symbol \rangle}{\langle insert \rangle}`

which is used subsequently. Both commands take four arguments: the first is either the singular or plural form given by the `text`, `plural`, `first` or `firstplural` keys (set when the term was defined) depending on context; the second argument is the term’s description (as supplied by the `description` or `descriptionplural` keys); the third argument is the symbol associated with the term (as supplied by the `symbol` or `symbolplural` keys) and the fourth argument is the additional text supplied in the final optional argument to `\gls` or `\glspl` (or their uppercase variants). The default definitions of `\glsdisplay` and `\glsdisplayfirst` simply print the first argument immediately followed by the fourth argument. The remaining arguments are ignored.

If required, you can access the label for the given entry via

`\glslabel` `\glslabel`

so it is possible to use this label in the definition of `\glsdisplay` or `\glsdisplayfirst` to supply additional information using any of the commands described in Section 2.7, if required.

Note that `\glsdisplay` and `\glsdisplayfirst` are not used by `\glslink`. If you want to supply your own link text, you need to use `\glsdisp` instead.

For example, suppose you want a glossary of measurements and units, you can use the `symbol` key to store the unit:

```
\newglossaryentry{distance}{name=distance,
description={The length between two points},
symbol={km}}
```

and now suppose you want `\gls{distance}` to produce “distance (km)” on first use, then you can redefine `\glsdisplayfirst` as follows:

```
\renewcommand{\glsdisplayfirst}[4]{#1#4 (#3)}
```

Note that the additional text is placed after `#1`, so `\gls{distance}[’s]` will produce “distance’s (km)” rather than “distance (km)’s” which looks a bit odd

(even though it may be in the context of “the distance (km) is measured between the two points” — but in this instance it would be better not to use a contraction).

Note also that all of the link text will be formatted according to `\glstextformat` (described earlier). So if you do, say:

```
\renewcommand{\glstextformat}[1]{\textbf{#1}}
\renewcommand{\glsdisplayfirst}[4]{#1#4 (#3)}
```

then `\gls{distance}` will produce “**distance (km)**”.

If you have multiple glossaries, changing `\glsdisplayfirst` and `\glsdisplay` will change the way entries for all of the glossaries appear when using the commands `\gls`, `\glspl`, their uppercase variants and `\glsdisp`. If you only want the change to affect entries for a given glossary, then you need to use

```
\defglsdisplay \defglsdisplay[⟨type⟩]{⟨definition⟩}
```

and

```
\defglsdisplayfirst \defglsdisplayfirst[⟨type⟩]{⟨definition⟩}
```

instead of redefining `\glsdisplay` and `\glsdisplayfirst`.

Both `\defglsdisplay` and `\defglsdisplayfirst` take two arguments: the first (which is optional) is the glossary’s label¹⁰ and the second is how the term should be displayed when it is invoked using commands `\gls`, `\glspl`, their uppercase variants and `\glsdisp`. This is similar to the way `\glsdisplayfirst` was redefined above.

For example, suppose you have created a new glossary called `notation` and you want to change the way the entry is displayed on first use so that it includes the symbol, you can do:

```
\defglsdisplayfirst[notation]{#1#4 (denoted #3)}
```

Now suppose you have defined an entry as follows:

```
\newglossaryentry{set}{type=notation,
  name=set,
  description={A collection of objects},
  symbol={$$}
}
```

The first time you reference this entry it will be displayed as: “set (denoted *S*)” (assuming `\gls` was used).

Remember that if you use the `symbol` key, you need to use a glossary style that displays the symbol, as many of the styles ignore it. In addition, if you want either the description or symbol to appear in the link text, you will have to disable the **sanitization** of these keys and protect fragile commands.

¹⁰`main` for the main (default) glossary, `acronymtype` for the list of acronyms, or the name supplied in the first mandatory argument to `\newglossary` for additional glossaries.

2.4.2 Enabling and disabling hyperlinks to glossary entries

If you load the `hyperref` or `html` packages prior to loading the `glossaries` package, commands such as `\glslink` and `\gls`, described above, will automatically have hyperlinks to the relevant glossary entry, unless the `hyper` option has been set to `false`. You can disable or enable links using:

`\glsdisablehyper`

```
\glsdisablehyper
```

and

`\glsenablehyper`

```
\glsenablehyper
```

respectively. The effect can be localised by placing the commands within a group. Note that you should only use `\glsenablehyper` if the commands `\hyperlink` and `\hypertarget` have been defined (for example, by the `hyperref` package).

You can disable just the first use links using the package option `hyperfirst=false`. Note that this option only affects commands that recognise the first use flag, for example `\gls`, `\glspl` and `\glsdisp` but not `\glslink`.

2.5 Adding an Entry to the Glossary Without Generating Text

It is possible to add a line in the glossary file without generating any text at that point in the document using:

`\glsadd`

```
\glsadd[⟨options⟩]{⟨label⟩}
```

This is similar to `\glslink`, only it doesn't produce any text (so therefore, there is no `hyper` key available in `⟨options⟩` but all the other options that can be used with `\glslink` can be passed to `\glsadd`). For example, to add a page range to the glossary number list for the entry whose label is given by `set`:

```
\glsadd[format=]{set}
Lots of text about sets spanning many pages.
\glsadd[format=)]{set}
```

To add all entries that have been defined, use:

`\glsaddall`

```
\glsaddall[⟨options⟩]
```

The optional argument is the same as for `\glsadd`, except there is also a key `types` which can be used to specify which glossaries to use. This should be a comma separated list. For example, if you only want to add all the entries belonging to the list of acronyms (specified by the glossary type `\acronymtype`) and a list of notation (specified by the glossary type `notation`) then you can do:

`\glsaddall[types={\acronymtype,notation}]`

2.6 Cross-Referencing Entries

There are several ways of cross-referencing entries in the glossary:

1. You can use commands such as `\gls` in the entries description. For example:

```
\newglossaryentry{apple}{name=apple,
description={firm, round fruit. See also \gls{pear}}}
```

Note that with this method, if you don't use the cross-referenced term in the glossary, you will need two runs of `makeglossaries`:

```
latex filename
makeglossaries filename
latex filename
makeglossaries filename
latex filename
```

2. As described in Section 2.2, you can use the `see` key when you define the entry. For example:

```
\newglossaryentry{MaclaurinSeries}{name={Maclaurin series},
description={Series expansion}, see={TaylorsTheorem}}
```

Note that in this case, the entry with the `see` key will automatically be added to the glossary, but the cross-referenced entry won't. You therefore need to ensure that you use the cross-referenced term with the commands described in Section 2.4 or Section 2.5.

You can optionally override the “see” tag using square brackets at the start of the `see` value. For example:

```
\newglossaryentry{MaclaurinSeries}{name={Maclaurin series},
description={Series expansion},
see=[see also]{TaylorsTheorem}}
```

3. After you have defined the entry, use

`\glssee`

`\glssee[<tag>]{<label>}{<xr label list>}`

where `<xr label list>` is a comma-separated list of entry labels to be cross-referenced, `<label>` is the label of the entry doing the cross-referencing and `<tag>` is the “see” tag. For example:

```
\glssee[see also]{series}{FourierSeries,TaylorsTheorem}
```


Note that this automatically adds the entry given by $\langle label \rangle$ to the glossary but doesn't add the cross-referenced entries (specified by $\langle xr label list \rangle$) to the glossary.

In both cases 2 and 3 above, the cross-referenced information appears in the number list, whereas in case 1, the cross-referenced information appears in the description. In cases 2 and 3, the default text for the “see” tag is given by `\seename`.

2.7 Using Glossary Terms Without Links

The commands described in this section display entry details without adding any information to the glossary. They don't use `\glstextformat`, they don't have any optional arguments, they don't affect the first use flag and, apart from `\glshyperlink`, they don't produce hyperlinks.

`\glentryname` `\glentryname{\langle label \rangle}`

`\Glentryname` `\Glentryname{\langle label \rangle}`

These commands display the name of the glossary entry given by $\langle label \rangle$, as specified by the `name` key. `\Glentryname` makes the first letter uppercase.

`\glentrytext` `\glentrytext{\langle label \rangle}`

`\Glentrytext` `\Glentrytext{\langle label \rangle}`

These commands display the subsequent use text for the glossary entry given by $\langle label \rangle$, as specified by the `text` key. `\Glentrytext` makes the first letter uppercase.

`\glentryplural` `\glentryplural{\langle label \rangle}`

`\Glentryplural` `\Glentryplural{\langle label \rangle}`

These commands display the subsequent use plural text for the glossary entry given by $\langle label \rangle$, as specified by the `plural` key. `\Glentryplural` makes the first letter uppercase.

`\glentryfirst` `\glentryfirst{\langle label \rangle}`

`\Glsentryfirst` `\Glsentryfirst{⟨label⟩}`

These commands display the first use text for the glossary entry given by *⟨label⟩*, as specified by the first key. `\Glsentryfirst` makes the first letter uppercase.

`\glsentryfirstplural` `\glsentryfirstplural{⟨label⟩}`

`\Glsentryfirstplural` `\Glsentryfirstplural{⟨label⟩}`

These commands display the plural form of the first use text for the glossary entry given by *⟨label⟩*, as specified by the `firstplural` key. `\Glsentryfirstplural` makes the first letter uppercase.

`\glsentrydesc` `\glsentrydesc{⟨label⟩}`

`\Glsentrydesc` `\Glsentrydesc{⟨label⟩}`

These commands display the description for the glossary entry given by *⟨label⟩*. `\Glsentrydesc` makes the first letter uppercase.

`\glsentrydescplural` `\glsentrydescplural{⟨label⟩}`

`\Glsentrydescplural` `\Glsentrydescplural{⟨label⟩}`

These commands display the plural description for the glossary entry given by *⟨label⟩*. `\Glsentrydescplural` makes the first letter uppercase.

`\glsentrysymbol` `\glsentrysymbol{⟨label⟩}`

`\Glsentrysymbol` `\Glsentrysymbol{⟨label⟩}`

These commands display the symbol for the glossary entry given by *⟨label⟩*. `\Glsentrysymbol` makes the first letter uppercase.

`\glsentrysymbolplural` `\glsentrysymbolplural{⟨label⟩}`

`\Glsentrysymbolplural`

`\Glsentrysymbolplural{<label>}`

These commands display the plural symbol for the glossary entry given by *<label>*.
`\Glsentrysymbolplural` makes the first letter uppercase.

`\glentryuseri`

`\glentryuseri{<label>}`

`\Glsentryuseri`

`\Glsentryuseri{<label>}`

`\glentryuserii`

`\glentryuserii{<label>}`

`\Glsentryuserii`

`\Glsentryuserii{<label>}`

`\glentryuseriii`

`\glentryuseriii{<label>}`

`\Glsentryuseriii`

`\Glsentryuseriii{<label>}`

`\glentryuseriv`

`\glentryuseriv{<label>}`

`\Glsentryuseriv`

`\Glsentryuseriv{<label>}`

`\glentryuserv`

`\glentryuserv{<label>}`

`\Glsentryuserv`

`\Glsentryuserv{<label>}`

`\glentryuservi`

`\glentryuservi{<label>}`

`\Glsentryuservi`

`\Glsentryuservi{<label>}`

These commands display the value of the user keys for the glossary entry given by $\langle label \rangle$.

`\glshyperlink` `\glshyperlink[$\langle link\ text \rangle$]{ $\langle label \rangle$ }`

This command provides a hyperlink to the glossary entry given by $\langle label \rangle$ **but does not add any information to the glossary file**. The link text is given by `\glseentryname{ $\langle label \rangle$ }` by default, but can be overridden using the optional argument.

If you use `\glshyperlink`, you need to ensure that the relevant entry has been added to the glossary using any of the commands described in Section 2.4 or Section 2.5 otherwise you will end up with a broken link.

For further information see Section 4.10.2.

2.8 Displaying a glossary

The command

`\printglossaries` `\printglossaries`

will display all the glossaries in the order in which they were defined. Note that no glossaries will appear until you have either used the Perl script `makeglossaries` or have directly used `makeindex` or `xindy` (as described in Section 1.3). If the glossary still does not appear after you re- \LaTeX your document, check the `makeindex/xindy` log files to see if there is a problem. Remember that you also need to use the command `\makeglossaries` in the preamble to enable the glossaries.

An individual glossary can be displayed using:

`\printglossary` `\printglossary[$\langle options \rangle$]`

where $\langle options \rangle$ is a $\langle key \rangle = \langle value \rangle$ list of options. The following keys are available:

type The value of this key specifies which glossary to print. If omitted, the default glossary is assumed. For example, to print the list of acronyms:

```
\printglossary[type=\acronymtype]
```

title This is the glossary's title (overriding the title specified when the glossary was defined).

toctitle This is the title to use for the table of contents (if the `toc` package option has been used). It may also be used for the page header, depending on the page style. If omitted, the glossary title is used.

style This specifies which glossary style to use for this glossary, overriding the effect of the `style` package option or `\glossarystyle`.

numberedsection This specifies whether to use a numbered section for this glossary, overriding the effect of the `numberedsection` package option. This key has the same syntax as the `numberedsection` package option, described in Section 2.1.

nonumberlist Unlike the package option of the same name, this key is a boolean key. If true (`nonumberlist=true`) the numberlist is suppressed for this glossary. If false (`nonumberlist=false`) the numberlist is displayed for this glossary. If no value is supplied, true is assumed.

By default, the glossary is started either by `\chapter*` or by `\section*`, depending on whether or not `\chapter` is defined. This can be overridden by the `section` package option or the `\setglossarysection` command. Numbered sectional units can be obtained using the `numberedsection` package option. Each glossary sets the page header via the command `\glossarymark`. If this mechanism is unsuitable for your chosen class file or page style package, you will need to redefine `\glossarymark`. Further information about these options and commands is given in Section 2.1.

Information can be added to the start of the glossary (after the title and before the main body of the glossary) by redefining

`\glossarypreamble`

`\glossarypreamble`

For example:

```
\renewcommand{\glossarypreamble}{Numbers in italic indicate
primary definitions.}
```

This needs to be done before the glossary is displayed using `\printglossaries` or `\printglossary`. Note that if you want a different preamble for each glossary, you will need to use a separate `\printglossary` for each glossary and change the definition of `\glossarypreamble` between each glossary. For example:

```
\renewcommand{\glossarypreamble}{Numbers in italic indicate
primary definitions.}
\printglossary
\renewcommand{\glossarypreamble}{}
\printglossary[type=acronym]
```

Alternatively, you can do something like:

```
\renewcommand{\glossarypreamble}{Numbers in italic indicate
primary definitions.\gdef\glossarypreamble{}}
\printglossaries
```

which will print the preamble text for the first glossary and change the preamble to do nothing for subsequent glossaries. (Note that `\gdef` is required as the glossary is placed within a group.)

There is an analogous command called

`\glossarypostamble`

`\glossarypostamble`

which is placed at the end of each glossary.

2.8.1 Changing the way the entry name appears in the glossary

Within each glossary, each entry name is formatted according to

`\glsnamefont`

`\glsnamefont{<name>}`

which takes one argument: the entry name. This command is always used regardless of the glossary style. By default, `\glsnamefont` simply displays its argument in whatever the surrounding font happens to be. This means that in the list-like glossary styles (defined in the `glossary-list` style file) the name will appear in bold, since the name is placed in the optional argument of `\item`, whereas in the tabular styles (defined in the `glossary-long` and `glossary-super` style files) the name will appear in the normal font. The hierarchical glossary styles (defined in the `glossary-tree` style file) also set the name in bold.

For example, suppose you want all the entry names to appear in medium weight small caps, then you can do:

```
\renewcommand{\glsnamefont}[1]{\textsc{\mdseries #1}}
```

2.8.2 Xindy

If you want to use `xindy` to sort the glossary, you must use the package option `xindy`:

```
\usepackage[xindy]{glossaries}
```

This ensures that the glossary information is written in `xindy` syntax.

Section 1.3 covers how to use the external indexing application. This section covers the commands provided by the `glossaries` package that allow you to adjust the `xindy` style file (`.xdy`) and parameters.

To assist writing information to the `xindy` style file, the `glossaries` package provides the following commands:

`\glsopenbrace`

`\glsopenbrace`

`\glsclosebrace`

`\glsclosebrace`

which produce an open and closing brace. (This is needed because `\{` and `\}` don't expand to a simple brace character when written to a file.)

In addition, if you are using a package that makes the double quote character active (e.g. `ngerman`) you can use:

`\glsquote`

```
\glsquote{<text>}
```

which will produce "`<text>`". Alternatively, you can use `\string` to write the double-quote character. This document assumes that the double quote character has not been made active, so the examples just use `"` for clarity.

If you want greater control over the `xindy` style file than is available through the \LaTeX commands provided by the `glossaries` package, you will need to edit the `xindy` style file. In which case, you must use `\noist` to prevent the style file from being overwritten by the `glossaries` package. For additional information about `xindy`, read the `xindy` documentation.

Language and Encodings When you use `xindy`, you need to specify the language and encoding used (unless you have written your own custom `xindy` style file that defines the relevant alphabet and sort rules). If you use `makeglossaries`, this information is obtained from the document's auxiliary (`.aux`) file. The `glossaries` package attempts to find the root language, but in the event that it gets it wrong or if `xindy` doesn't support that language, then you can specify the language using:

`\GlsSetXdyLanguage`

```
\GlsSetXdyLanguage[<glossary type>]{<language>}
```

where `<language>` is the name of the language. The optional argument can be used if you have multiple glossaries in different languages. If `<glossary type>` is omitted, it will be applied to all glossaries, otherwise the language setting will only be applied to the glossary given by `<glossary type>`.

If the `inputenc` package is used, the encoding will be obtained from the value of `\inputencodingname`. Alternatively, you can specify the encoding using:

`\GlsSetXdyCodePage`

```
\GlsSetXdyCodePage{<code>}
```

where `<code>` is the name of the encoding. For example:

```
\GlsSetXdyCodePage{utf8}
```

Note that you can also specify the language and encoding using the package option `xindy={language=<lang>,codepage=<code>}`. For example:

```
\usepackage[xindy={language=english,codepage=utf8}]{glossaries}
```

If you write your own custom `xindy` style file that includes the language settings, you need to set the language to nothing:

```
\GlsSetXdyLanguage{}
```

(and remember to use `\noist` to prevent the style file from being overwritten).

The commands `\GlsSetXdyLanguage` and `\GlsSetXdyCodePage` have no effect if you don't use `makeglossaries`. If you call `xindy` without `makeglossaries` you need to remember to set the language and encoding using the `-L` and `-C` switches.

Locations and Number lists The most likely attributes used in the format key (`\textrm`, `\hyperrm` etc) are automatically added to the `xindy` style file, but if you want to use another attribute, you need to add it using:

`\GlsAddXdyAttribute`

`\GlsAddXdyAttribute{<name>}`

where `<name>` is the name of the attribute, as used in the format key. For example, suppose I want a bold, italic, hyperlinked location. I first need to define a command that will do this:

```
\newcommand*{\hyperbfit}[1]{\textit{\hyperbf{#1}}}
```

but with `xindy`, I also need to add this as an allowed attribute:

```
\GlsAddXdyAttribute{hyperbfit}
```

Note that `\GlsAddXdyAttribute` has no effect if `\noist` is used or if `\makeglossaries` is omitted.
`\GlsAddXdyAttribute` must be used before `\makeglossaries`.

If the location numbers don't get expanded to a simple Arabic or Roman number or a letter from a, ..., z or A, ..., Z, then you need to add a location style in the appropriate format.

For example, suppose you want the page numbers written as words rather than digits and you use the `fmtcount` package to do this. You can redefine `\thepage` as follows:

```
\renewcommand*{\thepage}{\Numberstring{page}}
```

This gets expanded to `\protect \Numberstringnum {<n>}` where `<n>` is the Arabic page number. This means that you need to define a new location that has that form:

```
\GlsAddXdyLocation{Numberstring}{:sep "\string\protect\space  
  \string\Numberstringnum\space\glsopenbrace"  
  "arabic-numbers" :sep "\glsclosebrace"}
```


Note that it's necessary to use `\space` to indicate that spaces also appear in the format, since, unlike \TeX , `xindy` doesn't ignore spaces after control sequences.

Note that `\GlsAddXdyLocation` has no effect if `\noist` is used or if `\makeglossaries` is omitted.
`\GlsAddXdyLocation` must be used before `\makeglossaries`.

In the number list, the locations are sorted according to type. The default ordering is: `roman-page-numbers` (e.g. i), `arabic-page-numbers` (e.g. 1), `arabic-section-numbers` (e.g. 1.1 if the compositor is a full stop or 1-1 if the compositor is a hyphen¹¹), `alpha-page-numbers` (e.g. a), `Roman-page-numbers` (e.g. I), `Alpha-page-numbers` (e.g. A), `Appendix-page-numbers` (e.g. A.1 if the Alpha compositor is a full stop or A-1 if the Alpha compositor is a hyphen¹²), user defined location names (as specified by `\GlsAddXdyLocation` in the order in which they were defined), `see` (cross-referenced entries). This ordering can be changed using:

`\GlsSetXdyLocationClassOrder` `\GlsSetXdyLocationClassOrder{<location names>}`

where each location name is delimited by double quote marks and separated by white space. For example:

```
\GlsSetXdyLocationClassOrder{
  "arabic-page-numbers"
  "arabic-section-numbers"
  "roman-page-numbers"
  "Roman-page-numbers"
  "alpha-page-numbers"
  "Alpha-page-numbers"
  "Appendix-page-numbers"
  "see"
}
```

Note that `\GlsSetXdyLocationClassOrder` has no effect if `\noist` is used or if `\makeglossaries` is omitted.
`\GlsSetXdyLocationClassOrder` must be used before `\makeglossaries`.

If a number list consists of a sequence of consecutive numbers, the range will be concatenated. The number of consecutive locations that causes a range formation defaults to 2, but can be changed using:

`\GlsSetXdyMinRangeLength` `\GlsSetXdyMinRangeLength{<n>}`

¹¹see `\setCompositor` described in Section 2.2

¹²see `\setAlphaCompositor` described in Section 2.2

For example:

```
\GlsSetXdyMinRangeLength{3}
```

The argument may also be the keyword `none`, to indicate that there should be no range formations. See the `xindy` manual for further details on range formations.

Note that `\GlsSetXdyMinRangeLength` has no effect if `\noist` is used or if `\makeglossaries` is omitted.
`\GlsSetXdyMinRangeLength` must be used before `\makeglossaries`.

See Section 2.3 for further details.

Glossary Groups The glossary is divided into groups according to the first letter of the sort key. The `glossaries` package also adds a number group by default, unless you suppress it in the `xindy` package option. For example:

```
\usepackage[xindy={glsnumbers=false}]{glossaries}
```

Any entry that doesn't go in one of the letter groups or the number group is placed in the default group.

If you have a number group, the default behaviour is to locate it before the “A” letter group. If you are not using a Roman alphabet, you can change this using

```
\GlsSetXdyFirstLetterAfterDigits{<letter>}
```

Note that `\GlsSetXdyFirstLetterAfterDigits` has no effect if `\noist` is used or if `\makeglossaries` is omitted.
`\GlsSetXdyFirstLetterAfterDigits` must be used before `\makeglossaries`.

2.9 Defining New Glossaries

A new glossary can be defined using:

```
\newglossary    \newglossary[<log-ext>]{<name>}{<in-ext>}{<out-ext>}{<title>}[<counter>]
```

where `<name>` is the label to assign to this glossary. The arguments `<in-ext>` and `<out-ext>` specify the extensions to give to the input and output files for that glossary, `<title>` is the default title for this new glossary and the final optional argument `<counter>` specifies which counter to use for the associated number lists (see also Section 2.3). The first optional argument specifies the extension for the `makeindex` or `xindy` transcript file (this information is only used by `makeglossaries` which picks up the information from the auxiliary file).

Note that the main (default) glossary is automatically created as:

```
\newglossary{main}{gls}{glo}{\glossaryname}
```

so it can be identified by the label `main` (unless the `nomain` package option is used). Using the `acronym` package option is equivalent to:

```
\newglossary[alg]{acronym}{acr}{acn}{\acronymname}
```

`\acronymtype` so it can be identified by the label `acronym`. If you are not sure whether the `acronym` option has been used, you can identify the list of acronyms by the command `\acronymtype` which is set to `acronym`, if the `acronym` option has been used, otherwise it is set to `main`. Note that if you are using the main glossary as your list of acronyms, you need to declare it as a list of acronyms using the package option `acronymlists`.

All glossaries must be defined before `\makeglossaries` to ensure that the relevant output files are opened.

2.10 Acronyms

You may have noticed in Section 2.2 that when you specify a new entry, you can specify alternate text to use when the term is first used in the document. This provides a useful means to define acronyms. For convenience, the `glossaries` package defines the command:

`\newacronym` `\newacronym[<key-val list>]{<label>}{<abbrv>}{<long>}`

By default, this is equivalent to:

```
\newglossaryentry{<label>}{type=\acronymtype,
name={<abbrv>},
description={<long>},
text={<abbrv>},
first={<long> (<abbrv>)},
plural={<abbrv>\glspluralsuffix},
firstplural={<long>\glspluralsuffix\space (<abbrv>\glspluralsuffix)},
<key-val list>}
```

As mentioned in the previous section, the command `\acronymtype` is the name of the glossary in which the acronyms should appear. If the `acronym` package option has been used, this will be `acronym`, otherwise it will be `main`. The acronyms can then be used in exactly the same way as any other glossary entry. If you want more than one list of acronyms, you must identify the others using the package options `acronymlists`. This ensures that options such as `footnote` and `smallcaps` work for the additional lists of acronyms.

Note: since `\newacronym` sets `type=\acronymtype`, if you want to load a file containing acronym definitions using `\loadglsentries[⟨type⟩]{⟨filename⟩}`, the optional argument `⟨type⟩` will not have an effect unless you explicitly set the type as `type=\glsdefaulttype` in the optional argument to `\newacronym`. See Section 2.2.3.

For example, the following defines the acronym IDN:

```
\newacronym{idn}{IDN}{identification number}
```

This is equivalent to:

```
\newglossaryentry{idn}{type=\acronymtype,
name={IDN},
description={identification number},
text={IDN},
first={identification number (IDN)},
plural={IDNs},
firstplural={identification numbers (IDNs)}}
```

so `\gls{idn}` will produce “identification number (IDN)” on first use and “IDN” on subsequent uses.

This section describes acronyms that have been defined using `\newacronym`. If you prefer to define your acronyms using `\newglossaryentry` explicitly, then you should skip this section and ignore the package options: `smallcaps`, `smaller`, `description`, `dua` and `footnote`, as these options change the definition of `\newacronym` for common acronym formats as well as the way that the link text is displayed (see Section 2.4.1). Likewise you should ignore the package option `shortcuts` and the new commands described in this section, such as `\acrshort`, as they vary according to the definition of `\newacronym`.

If you want to define your own custom acronym style, see Section 2.10.1.

If you try using `\newglossaryentry` for entries in a designated list of acronyms in combination with any of the above named package options you are likely to get unexpected results such as empty brackets or empty footnotes.

If you use any of the package options `smallcaps`, `smaller`, `description` or `footnote`, the acronyms will be displayed in the document using:

```
\acronymfont    \acronymfont{⟨text⟩}
```

and

```
\firstacronymfont    \firstacronymfont{⟨text⟩}
```

where `\firstacronymfont` is applied on first use and `\acronymfont` is applied on subsequent use. Note that if you don’t use any of the above package options, changing the definition of `\acronymfont` or `\firstacronymfont` will have

no effect. In this case, the recommended route is to use either the `smaller` or the `smallcaps` package option and redefine `\acronymfont` and `\firstacronymfont` as required. (The `smallcaps` option sets the default plural suffix in an upright font to cancel the effect of `\textsc`, whereas `smaller` sets the suffix in the surrounding font.) For example, if you want acronyms in a normal font on first use and emphasized on subsequent use, do:

```
\usepackage[smaller]{glossaries}
\renewcommand*{\firstacronymfont}[1]{#1}
\renewcommand*{\acronymfont}[1]{\emph{#1}}
```

(Note that it is for this reason that the `relsize` package is not automatically loaded when selecting the `smaller` package option.)

Table 5 lists the package options that govern the acronym styles and how the `\newglossaryentry` keys are used to store `\langle long \rangle` (the long form) and `\langle abbrev \rangle` (the short form). Note that the `smallcaps` option redefines `\acronymfont` so that it sets its argument using `\textsc` (so you should use lower case characters in `\langle abbrev \rangle`) and the `smaller` option redefines `\acronymfont` to use `\textsmaller`,¹³ otherwise `\acronymfont` simply displays its argument in the surrounding font.

Table 5: Package options governing `\newacronym` and how the information is stored in the keys for `\newglossaryentry`

Package Option	first key	text key	description key	symbol key
description,footnote	<code>\langle abbrev \rangle</code>	<code>\langle abbrev \rangle</code>	user supplied	<code>\langle long \rangle</code>
description,dua	<code>\langle long \rangle</code>	<code>\langle long \rangle</code>	user supplied	<code>\langle abbrev \rangle</code>
description	<code>\langle long \rangle</code>	<code>\langle abbrev \rangle</code>	user supplied	<code>\langle abbrev \rangle</code>
footnote	<code>\langle abbrev \rangle</code>	<code>\langle abbrev \rangle</code>	<code>\langle long \rangle</code>	
smallcaps	<code>\langle long \rangle</code>	<code>\langle abbrev \rangle</code>	<code>\langle long \rangle</code>	<code>\langle abbrev \rangle</code>
smaller	<code>\langle long \rangle</code>	<code>\langle abbrev \rangle</code>	<code>\langle long \rangle</code>	<code>\langle abbrev \rangle</code>
dua	<code>\langle long \rangle</code>	<code>\langle long \rangle</code>	<code>\langle long \rangle</code>	<code>\langle abbrev \rangle</code>
None of the above	<code>\langle long \rangle (\langle abbrev \rangle)</code>	<code>\langle abbrev \rangle</code>	<code>\langle long \rangle</code>	

In case you can't remember which key stores the long or short forms (or their plurals) the `glossaries` package provides the commands:

<code>\glsshortkey</code>	• <code>\glsshortkey</code> The key used to store the short form.
<code>\glsshortpluralkey</code>	• <code>\glsshortpluralkey</code> The key used to store the plural version of the short form.
<code>\glslongkey</code>	• <code>\glslongkey</code> The key used to store the long form.
<code>\glslongpluralkey</code>	• <code>\glslongpluralkey</code> The key used to store the plural version of the long form.

¹³you will need to load a package, such as `relsize`, that defines `\textsmaller` if you use this option.

These can be used in the optional argument of `\newacronym` to override the defaults. For example:

```
\newacronym[\glslongpluralkey={diagonal matrices}]{dm}{DM}{diagonal
matrix}
```

If the first use uses the plural form, `\glspl{dm}` will display: diagonal matrices (DMs).

Each of the package options `smallcaps`, `smaller`, `footnote`, `dua` and `description` use `\defglsdisplay` and `\defglsdisplayfirst` (described in Section 2.4.1) to change the way the link text is displayed. This means that these package options only work for the glossary type given by `\acronymtype`. If you have multiple lists of acronyms, you will need to make the appropriate changes for each additional glossary type.

description,footnote

When these two package options are used together, the first use displays the entry as:

```
\firstacronymfont{<abbrv>}<insert>\footnote{<long>}
```

while subsequent use displays the entry as:

```
\acronymfont{<abbrv>}<insert>
```

where `<insert>` indicates the text supplied in the final optional argument to `\gls`, `\glspl` or their uppercase variants.

In this case, the long form is stored in the `symbol` key. This means that the long form will not be displayed in the list of acronyms unless you use a glossary style that displays the entry's symbol (for example, the `index` style). Entries will be sorted according to the short form.

Note also that when these two package options are used (in the given order), the `glossaries` package additionally implements the `sanitize` option using `sanitize={description=false,symbol=false}`, so remember to protect fragile commands when defining acronyms.

dua

The `dua` package option always displays the expanded form and so may not be used with `footnote`, `smaller` or `smallcaps`. Both first use and subsequent use displays the entry in the form:

```
<long><insert>
```

If the `description` package option is also used, the `name` key is set to the long form, otherwise the `name` key is set to the short form and the `description` key is set to the long form. In both cases the `symbol` is set to the short form. Therefore, if you use the `description` package option and you want the short form to appear in the list of acronyms, you will need to use a glossary style that displays the entry’s symbol (for example, the `index` style). Entries will be sorted according to the long form if the `description` option is used, otherwise they will be sorted according to the short form (unless overridden by the `sort` key in the optional argument of `\newacronym`).

description

This package option displays the entry on first use as:

```
<long><insert> (\firstacronymfont{<abbrv>})
```

while subsequent use displays the entry as:

```
\acronymfont{<abbrv>}<insert>
```

Note also that if this package option is used, the `glossaries` package additionally implements the option `sanitize={symbol=false}`, so remember to protect fragile commands when defining acronyms.

Note that with this option, you need to specify the description using the `description` key in the optional argument of `\newacronym`. When this option is used without the `footnote` or `dua` options, the `name` field is specified as

`\acrnameformat`

```
\acrnameformat{<short>}{<long>}
```

This defaults to `\acronymfont{<short>}`, which means that the long form will not appear in the list of acronyms by default. To change this, you need to redefine `\acrnameformat` as appropriate. For example, to display the long form followed by the short form in parentheses do:

```
\renewcommand*{\acrnameformat}[2]{#2 (\acronymfont{#1})}
```

Note that even if you redefine `\acrnameformat`, the entries will be sorted according to the short form, unless you override this using the `sort` key in the optional argument to `\newacronym`.

footnote

This package option displays the entry on first use as:

```
\firstacronymfont{<abbrv>}<insert>\footnote{<long>}
```

while subsequent use displays the entry as:

```
\acronymfont{\langle abbrev \rangle \langle insert \rangle}
```

Acronyms will be sorted according to the short form.

Note also that if this package option is used, the `glossaries` package additionally implements the option `sanitize={description=false}`, so remember to protect fragile commands when defining acronyms.

Note that on first use, it is the long form in the footnote that links to the relevant glossary entry (where hyperlinks are enabled), whereas on subsequent use, the acronym links to the relevant glossary entry. It is possible to change this to make the acronym on first use have the hyperlink instead of the footnote, but since the footnote marker will also be a hyperlink, you will have two hyperlinks in immediate succession. This can be ambiguous where the hyperlinks are coloured rather than boxed. The code required to change the first use to make the acronym a hyperlink is as follows:

```
\defglsdisplayfirst[\acronymtype]{%
\noexpand\protect\noexpand
\glslink[\@gls@link@opts]{\@gls@link@label}{\firstacronymfont{#1}#4}%
\noexpand\protect\noexpand\footnote{#2}}%
```

Note that this involves using internal commands (i.e. commands whose name contains an `@` character), so if this code is placed in a `.tex` file it needs to be placed within a `\makeatletter ... \makeatother` pair. (See <http://www.tex.ac.uk/cgi-bin/texfaq2html?label=atsigns> for further details.)

smallcaps

If neither the `footnote` nor `description` options have been set, this option displays the entry on first use as:

```
\langle long \rangle \langle insert \rangle (\firstacronymfont{\langle abbrev \rangle})
```

while subsequent use displays the entry as:

```
\acronymfont{\langle abbrev \rangle \langle insert \rangle}
```

where `\acronymfont` is set to `\textsc{#1}`.

Note that since the acronym is displayed using `\textsc`, the short form, `\langle abbrev \rangle`, should be specified in lower case. (Recall that `\textsc{abc}` produces ABC whereas `\textsc{ABC}` produces ABC.)

Note also that if this package option is used, the `glossaries` package additionally implements the option `sanitize={symbol=false}`, so remember to protect fragile commands when defining acronyms.

smaller

If neither the `footnote` nor `description` options have been set, this option displays the entry on first use as:

```
<long><insert> (\firstacronymfont{<abbrv>})
```

while subsequent use displays the entry as:

```
\acronymfont{<abbrv>}<insert>
```

where `\acronymfont` is set to `\textsmaller{#1}`.¹⁴ The entries will be sorted according to the short form.

Remember to load a package that defines `\textsmaller` (such as `relsize`) if you want to use this option, unless you want to redefine `\acronymfont` to use some other formatting command.

Note also that if this package option is used, the `glossaries` package additionally implements the option `sanitize={symbol=false}`, so remember to protect fragile commands when defining acronyms.

None of the above

If none of the package options `smallcaps`, `smaller`, `footnote`, `dua` or `description` are used, then on first use the entry is displayed as:

```
<long> (<abbrv>)<insert>
```

while subsequent use displays the entry as:

```
<abbrv><insert>
```

Entries will be sorted according to the short form. Note that if none of the acronym-related package options are used, the `sanitize` option will not be affected.

¹⁴not that this was change from using `\smaller` to `\textsmaller` as declarations cause a problem for `\makefirstuc`.

Recall from Section 2.4 that you can access the values of individual keys using commands like `\glstext`, so it is possible to use these commands to print just the long form or just the abbreviation without affecting the flag that determines whether the entry has been used. However the keys that store the long and short form vary depending on the acronym style, so the `glossaries` package provides commands that are set according to the package options. These are as follows:

`\acrshort` `\acrshort[<options>]{<label>}[<insert>]`

`\Acrshort` `\Acrshort[<options>]{<label>}[<insert>]`

`\ACRshort` `\ACRshort[<options>]{<label>}[<insert>]`

Print the abbreviated version with (if required) a hyperlink to the relevant entry in the glossary. This is usually equivalent to `\glstext` (or its uppercase variants) but may additionally put the link text within the argument to `\acronymfont`.

`\acrlong` `\acrlong[<options>]{<label>}[<insert>]`

`\Acrlong` `\Acrlong[<options>]{<label>}[<insert>]`

`\ACRlong` `\ACRlong[<options>]{<label>}[<insert>]`

Print the long version with (if required) a hyperlink to the relevant entry in the glossary. This is may be equivalent to `\glstdesc`, `\glssymbol` or `\glsfirst` (or their uppercase variants), depending on package options.

`\acrfull` `\acrfull[<options>]{<label>}[<insert>]`

`\Acrfull` `\Acrfull[<options>]{<label>}[<insert>]`

`\ACRfull` `\ACRfull[<options>]{<label>}[<insert>]`

Print the long version followed by the abbreviation in brackets with (if required) a hyperlink to the relevant entry in the glossary.

Note that if any of the above commands produce unexpected output and you haven't used any of the acronym-related package options, you will need to switch off the sanitization. For example:

```
\usepackage[sanitize=none]{glossaries}
```

However, if you do this, you must remember to protect fragile commands when defining acronyms or glossary entries.

Note that if you change the definition of `\newacronym`, you may additionally need to change the above commands as well as changing the way the text is displayed using `\defglsdisplay` and `\defglsdisplayfirst`.

The package option `shortcuts` provides the synonyms listed in [table 6](#). If any of those commands generate an “undefined control sequence” error message, check that you have enabled the shortcuts using the `shortcuts` package option. Note that there are no shortcuts for the commands that produce all upper case text.

Table 6: Synonyms provided by the package option `shortcuts`

Shortcut Command	Equivalent Command
<code>\acs</code>	<code>\acrshort</code>
<code>\Acs</code>	<code>\Acrshort</code>
<code>\acsp</code>	<code>\acrshortpl</code>
<code>\Acsp</code>	<code>\Acrshortpl</code>
<code>\acl</code>	<code>\acrlong</code>
<code>\Acl</code>	<code>\Acrlong</code>
<code>\aclp</code>	<code>\acrlongpl</code>
<code>\Aclp</code>	<code>\Acrlongpl</code>
<code>\acf</code>	<code>\acrfull</code>
<code>\Acf</code>	<code>\Acrfull</code>
<code>\acfp</code>	<code>\acrfullpl</code>
<code>\Acfp</code>	<code>\Acrfullpl</code>
<code>\ac</code>	<code>\gls</code>
<code>\Ac</code>	<code>\Gls</code>
<code>\acp</code>	<code>\glspl</code>
<code>\Acp</code>	<code>\Glspl</code>

2.10.1 Defining A Custom Acronym Style

You may find that the predefined acronyms styles that come with the `glossaries` package don't suit your requirements. In this case you can define your own style. This is done by redefining the following commands:

`\CustomAcronymFields`

```
\CustomAcronymFields
```

This command sets up the keys for `\newglossaryentry` when you define an acronym using `\newacronym`. Within the definition of `\CustomAcronymFields`, you may use `\the\glslongtok` to access the long form, `\the\glsshorttok` to access the short form and `\the\glslabeltok` to access the label. This command is typically used to set the name, first, firstplural, text and plural keys. It may also be used to set the symbol or description keys depending on your requirements.

`\SetCustomDisplayStyle` `\SetCustomDisplayStyle{<type>}`

This is used to set up the display style for the glossary given by `<type>`. This should typically just use `\defglsdisplayfirst` and `\defglsdisplay`.

Once you have redefined `\CustomAcronymFields` and `\SetCustomDisplayStyle`, you must then switch to this style using

`\SetCustomStyle` `\SetCustomStyle`

Note that you may still use the `shortcuts` package option with your custom style.

If you omit `\SetCustomStyle`, or use it before you redefine `\CustomAcronymFields` and `\SetCustomDisplayStyle`, your new style won't be correctly implemented. You must set up the custom style before defining new acronyms. The acronyms must be defined using `\newacronym` not `\newglossaryentry`.

As an example, suppose I want my acronym on first use to have the short form in the text and the long form with the description in a footnote. Suppose also that I want the short form to be put in small caps in the main body of the document, but I want it in normal capitals in the list of acronyms. In my list of acronyms, I want the long form as the name with the short form in brackets followed by the description.

First, I need to redefine `\CustomAcronymFields` so that `\newacronym` will correctly set the name, text and plural keys. I want the long form to be stored in the name and the short form to be stored in text. In addition, I'm going to set the symbol to the short form in upper case so that it will appear in the list of acronyms.

```
\renewcommand*{\CustomAcronymFields}{%
  name={\the\glslongtok},%
  symbol={\MakeUppercase{\the\glsshorttok}},%
  text={\textsc{\the\glsshorttok}},%
  plural={\textsc{\the\glsshorttok}s}%
}
```

Note that in this case I haven't bothered with `\acrpluralsuffix` and have just inserted an "s".

When I use the custom acronym style, the short form is stored in `user1`, the plural short form is stored in `user2`, the long form is stored in `user3` and the plural long form is stored in `user4`. So when I use `\defglsdisplayfirst` and `\defglsdisplay`, I can use `\glsentryuseriii` to access the long form. Recall from Section 2.4.1, that the optional argument to `\defglsdisplayfirst` and `\defglsdisplay` indicates the glossary type. This is passed to `\SetCustomDisplayStyle`. The mandatory argument sets up the definition of `\glsdisplayfirst` and `\glsdisplay` for the given glossary, where the first argument corresponds to the first, firstplural, text or plural, as appropriate, the second argument corresponds to the description, the third corresponds to the symbol and the fourth argument is the inserted text.

```
\renewcommand*{\SetCustomDisplayStyle}[1]{%
  \defglsdisplayfirst[#1]{##1##4\protect\footnote{%
    \glsentryuseriii{\glslabel}: ##2%
  }}
  \defglsdisplay[#1]{##1##4}%
}
```

Since we have a definition inside a definition, `#1` refers to the argument of `\SetCustomDisplayStyle`, and `##1`, ..., `##4`, refer to the arguments of `\glsdisplayfirst` and `\glsdisplay`.

Now that I've redefined `\CustomAcronymFields` and `\SetCustomDisplayStyle`, I can set this style using

```
\SetCustomStyle
```

and now I can define my acronyms:

```
\newacronym[description={set of tags for use in developing hypertext
documents}]{html}{html}{Hyper Text Markup Language}
```

```
\newacronym[description={language used to describe the layout of a
document written in a markup language}]{css}{css}{Cascading Style
Sheet}
```

Note that since I've used the description in the main body of the text, I need to switch off the sanitization otherwise any commands within the description won't get interpreted. Also I want to use the `hyperref` package, but this will cause a problem on first use as I'll get nested hyperlinks, so I need to switch off the hyperlinks on first use. In addition, I want to use a glossary style that displays the symbol. Therefore, in my preamble I have:

```
\usepackage[colorlinks]{hyperref}
\usepackage[acronym,          % create list of acronyms
             nomain,          % don't need main glossary for this example
             style=tree,      % need a style that displays the symbol
             hyperfirst=false,% don't hyperlink first use
             sanitize=none    % switch off sanitization as description
             % will be used in the main text
]{glossaries}
```

Note that I haven't used the `description` or `footnote` package options.

2.10.2 Upgrading From the `glossary` Package

Users of the obsolete `glossary` package may recall that the syntax used to define new acronyms has changed with the replacement `glossaries` package. In addition, the old `glossary` package created the command `\<acr-name>` when defining the acronym `<acr-name>`.

In order to facilitate migrating from the old package to the new one, the `glossaries` package¹⁵ provides the command:

```
\oldacronym[<label>]{<abbrv>}{<long>}{<key-val list>}
```

This uses the same syntax as the `glossary` package's method of defining acronyms. It is equivalent to:

```
\newacronym[<key-val list>]{<label>}{<abbrv>}{<long>}
```

In addition, `\oldacronym` also defines the commands `\<label>`, which is equivalent to `\gls{<label>}`, and `\<label>*`, which is equivalent to `\Gls{<label>}`. If `<label>` is omitted, `<abbrv>` is used. Since commands names must consist only of alphabetical characters, `<label>` must also only consist of alphabetical characters. Note that `\<label>` doesn't allow you to use the first optional argument of `\gls` or `\Gls` — you will need to explicitly use `\gls` or `\Gls` to change the settings.

Recall that, in general, L^AT_EX ignores spaces following command names consisting of alphabetical characters. This is also true for `\<label>` unless you additionally load the `xspace` package.

The `glossaries` package doesn't load the `xspace` package since there are both advantages and disadvantages to using `\xspace` in `\<label>`. If you don't use the `xspace` package you need to explicitly force a space using `_` (backslash space) however you can follow `\<label>` with additional text in square brackets (the final optional argument to `\gls`). If you use the `xspace` package you don't need to escape the spaces but you can't use the optional argument to insert text (you will have to explicitly use `\gls`).

To illustrate this, suppose I define the acronym “abc” as follows:

```
\oldacronym{abc}{example acronym}{}
```

This will create the command `\abc` and its starred version `\abc*`. Table 7 illustrates the effect of `\abc` (on subsequent use) according to whether or not the `xspace` package has been loaded. As can be seen from the final row in the table, the `xspace` package prevents the optional argument from being recognised.

¹⁵as from version 1.18

Table 7: The effect of using `xspace` with `\oldacronym`

Code	With <code>xspace</code>	Without <code>xspace</code>
<code>\abc.</code>	abc.	abc.
<code>\abc xyz</code>	abc xyz	abcxyz
<code>\abc\ xyz</code>	abc xyz	abc xyz
<code>\abc* xyz</code>	Abc xyz	Abc xyz
<code>\abc['s] xyz</code>	abc ['s] xyz	abc's xyz

2.11 Unsetting and Resetting Entry Flags

When using `\gls`, `\glspl` and their uppercase variants it is possible that you may want to use the value given by the first key, even though you have already used the glossary entry. Conversely, you may want to use the value given by the text key, even though you haven't used the glossary entry. The former can be achieved by one of the following commands:

`\glsreset` `\glsreset{<label>}`

`\glslocalreset` `\glslocalreset{<label>}`

while the latter can be achieved by one of the following commands:

`\glsunset` `\glsunset{<label>}`

`\glslocalunset` `\glslocalunset{<label>}`

You can also reset or unset all entries for a given glossary or list of glossaries using:

`\glsresetall` `\glsresetall[<glossary list>]`

`\glslocalresetall` `\glslocalresetall[<glossary list>]`

`\glsunsetall` `\glsunsetall[<glossary list>]`

`\glslocalunsetall` `\glslocalunsetall[<glossary list>]`

where *<glossary list>* is a comma-separated list of glossary labels. If omitted, all defined glossaries are assumed. For example, to reset all entries in the main glossary and the list of acronyms:

```
\glsresetall[main,acronym]
```

You can determine whether an entry's first use flag is set using:

```
\ifglsused
```

```
\ifglsused{<label>}{<true part>}{<false part>}
```

where *<label>* is the label of the required entry. If the entry has been used, *<true part>* will be done, otherwise *<false part>* will be done.

2.12 Glossary Styles

The `glossaries` package comes with some pre-defined glossary styles. Note that the styles are suited to different types of glossaries: some styles ignore the associated symbol; some styles are not designed for hierarchical entries, so they display sub-entries in the same way as they display top level entries; some styles are designed for homographs, so they ignore the name for sub-entries. You should therefore pick a style that suits your type of glossary. See [table 8](#) for a summary of the available styles.

The glossary style can be set using the `style` key in the optional argument to `\printglossary` or using the command:

```
\glossarystyle
```

```
\glossarystyle{<style-name>}
```

Some of the glossary styles may also be set using the `style` package option, it depends if the package in which they are defined is automatically loaded by the `glossaries` package.

```
\glsdescwidth  
\glspagelistwidth
```

The tabular-like styles that allow multi-line descriptions and page lists use the length `\glsdescwidth` to set the width of the description column and the length `\glspagelistwidth` to set the width of the page list column.¹⁶ These will need to be changed using `\setlength` if the glossary is too wide. Note that the `long4col` and `super4col` styles (and their header and border variations) don't use these lengths as they are designed for single line entries. Instead you should use the analogous `altlong4col` and `altsuper4col` styles. If you want to explicitly create a line-break within a multi-line description in a tabular-like style you should use `\newline` instead of `\\`.

Note that if you use the `style` key in the optional argument to `\printglossary`, it will override any previous style settings for the given glossary, so if, for example, you do

```
\renewcommand*{\glsgroupskip}{}  
\printglossary[style=long]
```

¹⁶these lengths will not be available if you use both the `nolong` and `nosuper` package options or if you use the `nostyles` package option unless you explicitly load the relevant package.

Table 8: Glossary Styles. An asterisk in the style name indicates anything that matches that doesn't match any previously listed style (e.g. `long3col*` matches `long3col`, `long3colheader`, `long3colborder` and `long3colheaderborder`). A maximum level of 0 indicates a flat glossary (sub-entries are displayed in the same way as main entries). Where the maximum level is given as — there is no limit, but note that `makeindex` imposes a limit of 2 sub-levels. If the homograph column is checked, then the name is not displayed for sub-entries. If the symbol column is checked, then the symbol will be displayed if it has been defined.

Style	Maximum Level	Homograph	Symbol
<code>listdotted</code>	0		
<code>sublistdotted</code>	1		
<code>list*</code>	1	✓	
<code>altlist*</code>	1	✓	
<code>long*3col*</code>	1	✓	
<code>long4col*</code>	1	✓	✓
<code>altlong*4col*</code>	1	✓	✓
<code>long*</code>	1	✓	
<code>super*3col*</code>	1	✓	
<code>super4col*</code>	1	✓	✓
<code>altsuper*4col*</code>	1	✓	✓
<code>super*</code>	1	✓	
<code>index*</code>	2		✓
<code>treenoname*</code>	—	✓	✓
<code>tree*</code>	—		✓
<code>alttree*</code>	—		✓

then the new definition of `\glsgroupskip` will not have an affect for this glossary, as `\glsgroupskip` is redefined by `style=long`. Likewise, `\glossarystyle` will also override any previous style definitions, so, again

```
\renewcommand*{\glsgroupskip}{}
\glossarystyle{long}
```

will reset `\glsgroupskip` back to its default definition for the named glossary style (`long` in this case). If you want to modify the styles, either use `\newglossarystyle` (described in the next section) or make the modifications after `\glossarystyle`, e.g.:

```
\glossarystyle{long}
\renewcommand*{\glsgroupskip}{}

```

All the styles except for the three- and four-column styles and the `listdotted` style use the command

`\glspostdescription`

`\glspostdescription`

after the description. This simply displays a full stop by default. To eliminate this full stop (or replace it with something else, say, a comma) you will need to redefine `\glspostdescription` before the glossary is displayed. Alternatively, you can suppress it for a given entry by placing `\nopostdesc` in the entry's description.

2.12.1 List Styles

The styles described in this section are all defined in the package `glossary-list`. Since they all use the `description` environment, they are governed by the same parameters as that environment. These styles all ignore the entry's symbol. Note that these styles will automatically be available unless you use the `nolist` or `nostyles` package options.

list The `list` style uses the `description` environment. The entry name is placed in the optional argument of the `\item` command (so it will appear in bold by default). The description follows, and then the associated number list for that entry. The symbol is ignored. If the entry has child entries, the description and number list follows (but not the name) for each child entry. Groups are separated using `\indexspace`.

listgroup The `listgroup` style is like `list` but the glossary groups have headings.

listhypergroup The `listhypergroup` style is like `listgroup` but has a navigation line at the start of the glossary with links to each group that is present in the glossary. This requires an additional run through `LATEX` to ensure the group information is up to date. In the navigation line, each group is separated by

`\glshypernavsep`

`\glshypernavsep`

which defaults to a vertical bar with a space on either side. For example, to simply have a space separating each group, do:

```
\renewcommand*{\glshypernavsep}{\space}
```

Note that the hyper-navigation line is now (as from version 1.14) set inside the optional argument to `\item` instead of after it to prevent a spurious space at the start. This can be changed by redefining `\glossaryheader`, but note that this needs to be done *after* the glossary style has been set.

altlist The `altlist` style is like `list` but the description starts on the line following the name. (As with the `list` style, the symbol is ignored.) Each child entry starts a new line, but as with the `list` style, the name associated with each child entry is ignored.

altlistgroup The `altlistgroup` style is like `altlist` but the glossary groups have headings.

altlisthypergroup The `altlisthypergroup` style is like `altlistgroup` but has a set of links to the glossary groups. The navigation line is the same as that for `listhypergroup`, described above.

listdotted This style uses the `description` environment.¹⁷ Each entry starts with `\item[]`, followed by the name followed by a dotted line, followed by the description. Note that this style ignores both the number list and the symbol. The length

`\glslistdottedwidth`

`\glslistdottedwidth`

governs where the description should start. This is a flat style, so child entries are formatted in the same way as the parent entries.

sublistdotted This is a variation on the `listdotted` style designed for hierarchical glossaries. The main entries have just the name displayed. The sub entries are displayed in the same manner as `listdotted`.

2.12.2 Longtable Styles

The styles described in this section are all defined in the package `glossary-long`. Since they all use the `longtable` environment, they are governed by the same parameters as that environment. Note that these styles will automatically be available unless you use the `nolong` or `nostyles` package options. These styles fully justify the description and page list columns. If you want ragged right formatting instead, use the analogous styles described in Section 2.12.3.

¹⁷This style was supplied by Axel Menzel.

- long** The `long` style uses the `longtable` environment (defined by the `longtable` package). It has two columns: the first column contains the entry's name and the second column contains the description followed by the number list. The entry's symbol is ignored. Sub groups are separated with a blank row. The width of the first column is governed by the widest entry in that column. The width of the second column is governed by the length `\glsdescwidth`. Child entries have a similar format to the parent entries except that their name is suppressed.
- longborder** The `longborder` style is like `long` but has horizontal and vertical lines around it.
- longheader** The `longheader` style is like `long` but has a header row.
- longheaderborder** The `longheaderborder` style is like `longheader` but has horizontal and vertical lines around it.
- long3col** The `long3col` style is like `long` but has three columns. The first column contains the entry's name, the second column contains the description and the third column contains the number list. The entry's symbol is ignored. The width of the first column is governed by the widest entry in that column, the width of the second column is governed by the length `\glsdescwidth`, and the width of the third column is governed by the length `\glspagelistwidth`.
- long3colborder** The `long3colborder` style is like the `long3col` style but has horizontal and vertical lines around it.
- long3colheader** The `long3colheader` style is like `long3col` but has a header row.
- long3colheaderborder** The `long3colheaderborder` style is like `long3colheader` but has horizontal and vertical lines around it.
- long4col** The `long4col` style is like `long3col` but has an additional column in which the entry's associated symbol appears. This style is used for brief single line descriptions. The column widths are governed by the widest entry in the given column. Use `altlong4col` for multi-line descriptions.
- long4colborder** The `long4colborder` style is like the `long4col` style but has horizontal and vertical lines around it.
- long4colheader** The `long4colheader` style is like `long4col` but has a header row.
- long4colheaderborder** The `long4colheaderborder` style is like `long4colheader` but has horizontal and vertical lines around it.
- altlong4col** The `altlong4col` style is like `long4col` but allows multi-line descriptions and page lists. The width of the description column is governed by the length `\glsdescwidth` and the width of the page list column is governed by the length `\glspagelistwidth`. The widths of the name and symbol columns are governed by the widest entry in the given column.

altlong4colborder The altlong4colborder style is like the long4colborder but allows multi-line descriptions and page lists.

altlong4colheader The altlong4colheader style is like long4colheader but allows multi-line descriptions and page lists.

altlong4colheaderborder The altlong4colheaderborder style is like long4colheaderborder but allows multi-line descriptions and page lists.

2.12.3 Longtable Styles (Ragged Right)

The styles described in this section are all defined in the package `glossary-longragged`. These styles are analogous to those defined in `glossary-long` but the multiline columns are left justified instead of fully justified. Since these styles all use the `longtable` environment, they are governed by the same parameters as that environment. The `glossary-longragged` package additionally requires the `array` package. Note that these styles will only be available if you explicitly load `glossary-longragged`:

```
\usepackage{glossaries}  
\usepackage{glossary-longragged}
```

Note that you can't set these styles using the `style` package option since the styles aren't defined until after the `glossaries` package has been loaded.

longragged The longragged style has two columns: the first column contains the entry's name and the second column contains the (left-justified) description followed by the number list. The entry's symbol is ignored. Sub groups are separated with a blank row. The width of the first column is governed by the widest entry in that column. The width of the second column is governed by the length `\glsdescwidth`. Child entries have a similar format to the parent entries except that their name is suppressed.

longraggedborder The longraggedborder style is like longragged but has horizontal and vertical lines around it.

longraggedheader The longraggedheader style is like longragged but has a header row.

longraggedheaderborder The longraggedheaderborder style is like longraggedheader but has horizontal and vertical lines around it.

longragged3col The longragged3col style is like longragged but has three columns. The first column contains the entry's name, the second column contains the (left justified) description and the third column contains the (left justified) number list. The entry's symbol is ignored. The width of the first column is governed by the widest entry in that column, the width of the second column is governed by the length `\glsdescwidth`, and the width of the third column is governed by the length `\glspagelistwidth`.

longragged3colborder The `longragged3colborder` style is like the `longragged3col` style but has horizontal and vertical lines around it.

longragged3colheader The `longragged3colheader` style is like `longragged3col` but has a header row.

longragged3colheaderborder The `longragged3colheaderborder` style is like `longragged3colheader` but has horizontal and vertical lines around it.

altlongragged4col The `altlongragged4col` style is like `longragged3col` but has an additional column in which the entry's associated symbol appears. The width of the description column is governed by the length `\glsdescwidth` and the width of the page list column is governed by the length `\glspagelistwidth`. The widths of the name and symbol columns are governed by the widest entry in the given column.

altlongragged4colborder The `altlongragged4colborder` style is like the `altlongragged4col` but has horizontal and vertical lines around it.

altlongragged4colheader The `altlongragged4colheader` style is like `altlongragged4col` but has a header row.

altlongragged4colheaderborder The `altlongragged4colheaderborder` style is like `altlongragged4colheader` but has horizontal and vertical lines around it.

2.12.4 Supertabular Styles

The styles described in this section are all defined in the package `glossary-super`. Since they all use the `supertabular` environment, they are governed by the same parameters as that environment. Note that these styles will automatically be available unless you use the `nosuper` or `nostyles` package options. In general, the `longtable` environment is better, but there are some circumstances where it is better to use `supertabular`.¹⁸ These styles fully justify the description and page list columns. If you want ragged right formatting instead, use the analogous styles described in Section 2.12.5.

super The `super` style uses the `supertabular` environment (defined by the `supertabular` package). It has two columns: the first column contains the entry's name and the second column contains the description followed by the number list. The entry's symbol is ignored. Sub groups are separated with a blank row. The width of the first column is governed by the widest entry in that column. The width of the second column is governed by the length `\glsdescwidth`. Child entries have a similar format to the parent entries except that their name is suppressed.

superborder The `superborder` style is like `super` but has horizontal and vertical lines around it.

¹⁸e.g. with the `flowfram` package.

superheader The `superheader` style is like `super` but has a header row.

superheaderborder The `superheaderborder` style is like `superheader` but has horizontal and vertical lines around it.

super3col The `super3col` style is like `super` but has three columns. The first column contains the entry's name, the second column contains the description and the third column contains the number list. The entry's symbol is ignored. The width of the first column is governed by the widest entry in that column. The width of the second column is governed by the length `\glsdescwidth`. The width of the third column is governed by the length `\glspagelistwidth`.

super3colborder The `super3colborder` style is like the `super3col` style but has horizontal and vertical lines around it.

super3colheader The `super3colheader` style is like `super3col` but has a header row.

super3colheaderborder The `super3colheaderborder` style is like `super3colheader` but has horizontal and vertical lines around it.

super4col The `super4col` style is like `super3col` but has an additional column in which the entry's associated symbol appears. This style is designed for entries with brief single line descriptions. The column widths are governed by the widest entry in the given column. Use `altsuper4col` for multi-line descriptions.

super4colborder The `super4colborder` style is like the `super4col` style but has horizontal and vertical lines around it.

super4colheader The `super4colheader` style is like `super4col` but has a header row.

super4colheaderborder The `super4colheaderborder` style is like `super4colheader` but has horizontal and vertical lines around it.

altsuper4col The `altsuper4col` style is like `super4col` but allows multi-line descriptions and page lists. The width of the description column is governed by the length `\glsdescwidth` and the width of the page list column is governed by the length `\glspagelistwidth`. The width of the name and symbol columns is governed by the widest entry in the given column.

altsuper4colborder The `altsuper4colborder` style is like the `super4colborder` style but allows multi-line descriptions and page lists.

altsuper4colheader The `altsuper4colheader` style is like `super4colheader` but allows multi-line descriptions and page lists.

altsuper4colheaderborder The `altsuper4colheaderborder` style is like `super4colheaderborder` but allows multi-line descriptions and page lists.

2.12.5 Supertabular Styles (Ragged Right)

The styles described in this section are all defined in the package `glossary-superragged`. These styles are analogous to those defined in `glossary-super` but the multiline columns are left justified instead of fully justified. Since these styles all use the `supertabular` environment, they are governed by the same parameters as that environment. The `glossary-superragged` package additionally requires the `array` package. Note that these styles will only be available if you explicitly load `glossary-superragged`:

```
\usepackage{glossaries}
\usepackage{glossary-superragged}
```

Note that you can't set these styles using the `style` package option since the styles aren't defined until after the `glossaries` package has been loaded.

superragged The `superragged` style uses the `supertabular` environment (defined by the `supertabular` package). It has two columns: the first column contains the entry's name and the second column contains the (left justified) description followed by the number list. The entry's symbol is ignored. Sub groups are separated with a blank row. The width of the first column is governed by the widest entry in that column. The width of the second column is governed by the length `\glsdescwidth`. Child entries have a similar format to the parent entries except that their name is suppressed.

superraggedborder The `superraggedborder` style is like `superragged` but has horizontal and vertical lines around it.

superraggedheader The `superraggedheader` style is like `superragged` but has a header row.

superraggedheaderborder The `superraggedheaderborder` style is like `superraggedheader` but has horizontal and vertical lines around it.

superragged3col The `superragged3col` style is like `superragged` but has three columns. The first column contains the entry's name, the second column contains the (left justified) description and the third column contains the (left justified) number list. The entry's symbol is ignored. The width of the first column is governed by the widest entry in that column. The width of the second column is governed by the length `\glsdescwidth`. The width of the third column is governed by the length `\glspagelistwidth`.

superragged3colborder The `superragged3colborder` style is like the `superragged3col` style but has horizontal and vertical lines around it.

superragged3colheader The `superragged3colheader` style is like `superragged3col` but has a header row.

superragged3colheaderborder The `superragged3colheaderborder` style is like `superragged3colheader` but has horizontal and vertical lines around it.

altsuperragged4col The `altsuperragged4col` style is like `superragged3col` but has an additional column in which the entry’s associated symbol appears. The column widths for the name and symbol column are governed by the widest entry in the given column.

altsuperragged4colborder The `altsuperragged4colborder` style is like the `altsuperragged4col` style but has horizontal and vertical lines around it.

altsuperragged4colheader The `altsuperragged4colheader` style is like `altsuperragged4col` but has a header row.

altsuperragged4colheaderborder The `altsuperragged4colheaderborder` style is like `altsuperragged4colheader` but has horizontal and vertical lines around it.

2.12.6 Tree-Like Styles

The styles described in this section are all defined in the package `glossary-tree`. These styles are designed for hierarchical glossaries but can also be used with glossaries that don’t have sub-entries. These styles will display the entry’s symbol if it exists. Note that these styles will automatically be available unless you use the `notree` or `nostyles` package options.

index The `index` style is similar to the way indices are usually formatted in that it has a hierarchical structure up to three levels (the main level plus two sub-levels). The name is typeset in bold, and if the symbol is present it is set in parentheses after the name and before the description. Sub-entries are indented and also include the name, the symbol in brackets (if present) and the description. Groups are separated using `\indexspace`.

indexgroup The `indexgroup` style is similar to the `index` style except that each group has a heading.

indexhypergroup The `indexhypergroup` style is like `indexgroup` but has a set of links to the glossary groups. The navigation line is the same as that for `listhypergroup`, described above.

tree The `tree` style is similar to the `index` style except that it can have arbitrary levels. (Note that `makeindex` is limited to three levels, so you will need to use `xindy` if you want more than three levels.) Each sub-level is indented by `\glstreeindent`. Note that the name, symbol (if present) and description are placed in the same paragraph block. If you want the name to be apart from the description, use the `alttree` style instead. (See below.)

treegroup The `treegroup` style is similar to the `tree` style except that each group has a heading.

treehypergroup The `treehypergroup` style is like `treegroup` but has a set of links to the glossary groups. The navigation line is the same as that for `listhypergroup`, described above.

treenoname The `treenoname` style is like the `tree` style except that the name for each sub-entry is ignored.

treenonamegroup The `treenonamegroup` style is similar to the `treenoname` style except that each group has a heading.

treenonamehypergroup The `treenonamehypergroup` style is like `treenonamegroup` but has a set of links to the glossary groups. The navigation line is the same as that for `listhypergroup`, described above.

alttree The `alttree` style is similar to the `tree` style except that the indentation for each level is determined by the width of the text specified by

`\glssetwidest`

```
\glssetwidest[⟨level⟩]{⟨text⟩}
```

The optional argument `⟨level⟩` indicates the level, where 0 indicates the top-most level, 1 indicates the first level sub-entries, etc. If `\glssetwidest` hasn't been used for a given sub-level, the level 0 widest text is used instead. If `⟨level⟩` is omitted, 0 is assumed.

For each level, the name is placed to the left of the paragraph block containing the symbol (optional) and the description. If the symbol is present, it is placed in parentheses before the description.

alttreegroup The `alttreegroup` is like the `alttree` style except that each group has a heading.

alttreehypergroup The `alttreehypergroup` style is like `alttreegroup` but has a set of links to the glossary groups. The navigation line is the same as that for `listhypergroup`, described above.

2.13 Defining your own glossary style

If the predefined styles don't fit your requirements, you can define your own style using:

`\newglossarystyle`

```
\newglossarystyle{⟨name⟩}{⟨definitions⟩}
```

where `⟨name⟩` is the name of the new glossary style (to be used in `\glossarystyle`). The second argument `⟨definitions⟩` needs to redefine all of the following:

`theglossary`

```
theglossary
```

This environment defines how the main body of the glossary should be typeset. Note that this does not include the section heading, the glossary preamble (defined by `\glossarypreamble`) or the glossary postamble (defined by

`\glossarypostamble`). For example, the list style uses the `description` environment, so the `theglossary` environment is simply redefined to begin and end the `description` environment.

`\glossaryheader`

`\glossaryheader`

This macro indicates what to do at the start of the main body of the glossary. Note that this is not the same as `\glossarypreamble`, which should not be affected by changes in the glossary style. The list glossary style redefines `\glossaryheader` to do nothing, whereas the `longheader` glossary style redefines `\glossaryheader` to do a header row.

`\glsgroupheading`

`\glsgroupheading{<label>}`

This macro indicates what to do at the start of each logical block within the main body of the glossary. If you use `makeindex` the glossary is sub-divided into a maximum of twenty-eight logical blocks that are determined by the first character of the `sort` key (or `name` key if the `sort` key is omitted). The sub-divisions are in the following order: symbols, numbers, A, . . . , Z. If you use `xindy`, the sub-divisions depend on the language settings.

Note that the argument to `\glsgroupheading` is a label *not* the group title. The group title can be obtained via

`\glsgrouptitle`

`\glsgrouptitle{<label>}`

This obtains the title as follows: if `\<label>groupname` exists, this is taken to be the title, otherwise the title is just `<label>`.

A navigation hypertarget can be created using

`\glsnahypertarget`

`\glsnahypertarget{<label>}{<text>}`

For further details about `\glsnahypertarget`, see Section 6.1.

Most of the predefined glossary styles redefine `\glsgroupheading` to simply ignore its argument. The `listhypergroup` style redefines `\glsgroupheading` as follows:

```
\renewcommand*{\glsgroupheading}[1]{%
\item[\glsnahypertarget{##1}{\glsgrouptitle{##1}}]}
```

See also `\glsgroupskip` below. (Note that command definitions within `\newglossarystyle` must use `##1` instead of `#1` etc.)

`\glsgroupskip`

`\glsgroupskip`

This macro determines what to do after one logical group but before the header for the next logical group. The list glossary style simply redefines `\glsgroupskip`

to be `\indexspace`, whereas the tabular-like styles redefine `\glsgroupskip` to produce a blank row.

`\glossaryentryfield`

```
\glossaryentryfield{<label>}{<formatted name>}{<description>}{<symbol>}{<number list>}
```

This macro indicates what to do for a given glossary entry. Note that `<formatted name>` will always be in the form `\glsnamefont{<name>}`. This allows the user to set a given font for the entry name, regardless of the glossary style used. Note that `<label>` is the label used when the glossary entry was defined via either `\newglossaryentry` or `\newacronym`.

Each time you use a glossary entry it creates a hyperlink (if hyperlinks are enabled) to the relevant line in the glossary. Your new glossary style must therefore redefine `\glossaryentryfield` to set the appropriate target. This is done using

`\glstarget`

```
\glstarget{<label>}{<text>}
```

where `<label>` is the entry's label. Note that you don't need to worry about whether the `hyperref` package has been loaded, as `\glstarget` won't create a target if `\hypertarget` hasn't been defined.

For example, the list style defines `\glossaryentryfield` as follows:

```
\renewcommand*{\glossaryentryfield}[5]{%
\item[\glstarget{##1}{##2}] ##3\glspostdescription\space ##5}
```

Note also that `<number list>` will always be of the form

```
\glossaryentrynumbers{\relax
\setentrycounter{<counter name>}\glsnumberformat{<number(s)>}}
```

where `<number(s)>` may contain `\delimN` (to delimit individual numbers) and/or `\delimR` (to indicate a range of numbers). There may be multiple occurrences of `\setentrycounter{<counter name>}\glsnumberformat{<number(s)>}`, but note that the entire number list is enclosed within the argument to `\glossaryentrynumbers`. The user can redefine this to change the way the entire number list is formatted, regardless of the glossary style. However the most common use of `\glossaryentrynumbers` is to provide a means of suppressing the number list altogether. (In fact, the `nonumberlist` option redefines `\glossaryentrynumbers` to ignore its argument.) Therefore, when you define a new glossary style, you don't need to worry about whether the user has specified the `nonumberlist` package option.

`\glossarysubentryfield`

```
\glossarysubentryfield{<level>}{<label>}{<formatted
name>}{<description>}{<symbol>}{<number list>}
```

This is new to version 1.17, and is used to display sub-entries. The first ar-

gument, $\langle level \rangle$, indicates the sub-entry level. This must be an integer from 1 (first sub-level) onwards. The remaining arguments are analogous to those for `\glossaryentryfield` described above.

For further details of these commands, see Section 4.15.

2.13.1 Example: creating a completely new style

If you want a completely new style, you will need to redefine all of the commands and the environment listed above.

For example, suppose you want each entry to start with a bullet point. This means that the glossary should be placed in the `itemize` environment, so `theglossary` should start and end that environment. Let's also suppose that you don't want anything between the glossary groups (so `\glsgroupheading` and `\glsgroupskip` should do nothing) and suppose you don't want anything to appear immediately after `\begin{theglossary}` (so `\glossaryheader` should do nothing). In addition, let's suppose the symbol should appear in brackets after the name, followed by the description and last of all the number list should appear within square brackets at the end. Then you can create this new glossary style, called, say, `mylist`, as follows:

```
\newglossarystyle{mylist}{%
% put the glossary in the itemize environment:
\renewenvironment{theglossary}{\begin{itemize}}{\end{itemize}}%
% have nothing after \begin{theglossary}:
\renewcommand*{\glossaryheader}{}%
% have nothing between glossary groups:
\renewcommand*{\glsgroupheading}[1]{}%
\renewcommand*{\glsgroupskip}{}%
% set how each entry should appear:
\renewcommand*{\glossaryentryfield}[5]{%
\item % bullet point
\glstarget{##1}{##2}% the entry name
\space (##4)% the symbol in brackets
\space ##3% the description
\space [##5]% the number list in square brackets
}%
% set how sub-entries appear:
\renewcommand*{\glossarysubentryfield}[6]{%
\glossaryentryfield{##2}{##3}{##4}{##5}{##6}}%
}
```

Note that this style creates a flat glossary, where sub-entries are displayed in exactly the same way as the top level entries.

2.13.2 Example: creating a new glossary style based on an existing style

If you want to define a new style that is a slightly modified version of an existing style, you can use `\glossarystyle` within the second argument of

`\newglossarystyle` followed by whatever alterations you require. For example, suppose you want a style like the `list` style but you don't want the extra vertical space created by `\indexspace` between groups, then you can create a new glossary style called, say, `mylist` as follows:

```
\newglossarystyle{mylist}{%
\glossarystyle{list}% base this style on the list style
\renewcommand{\glsgroupskip}{}% make nothing happen between groups
}
```

2.13.3 Example: creating a glossary style that uses the `user1`, ..., `user6` keys

Since `\glossaryentryfield` and `\glossarysubentryfield` provide the label for the entry, it's also possible to access the values of the generic user keys, such as `user1`. For example, suppose each entry not only has an associated symbol, but also units (stored in `user1`) and dimension (stored in `user2`). Then you can define a glossary style that displays each entry in a `longtable` as follows:

```
\newglossarystyle{long6col}{%
% put the glossary in a longtable environment:
\renewenvironment{theglossary}%
{\begin{longtable}{lp{\glsgdescwidth}cccp{\glspagelistwidth}}}%
{\end{longtable}}%
% Set the table's header:
\renewcommand*{\glossaryheader}{%
\bfseries Term & \bfseries Description & \bfseries Symbol &
\bfseries Units & \bfseries Dimensions & \bfseries Page List
\\endhead}%
% No heading between groups:
\renewcommand*{\glsgroupheading}[1]{%
% Main (level 0) entries displayed in a row:
\renewcommand*{\glossaryentryfield}[5]{%
\glstarget{##1}{##2}% Name
& ##3% Description
& ##4% Symbol
& \glentryuseri{##1}% Units
& \glentryuserii{##1}% Dimensions
& ##5% Page list
\\% end of row
}%
% Sub entries treated the same as level 0 entries:
\renewcommand*{\glossarysubentryfield}[6]{%
\glossaryentryfield{##2}{##3}{##4}{##5}{##6}}%
% Nothing between groups:
\renewcommand*{\glsgroupskip}{}%
}
```

2.14 Accessibility Support

Limited accessibility support is provided by the accompanying `glossaries-accsupp` package, but note that this package is experimental and it requires the `accsupp` package which is also listed as experimental. This package defines additional keys that may be used when defining glossary entries. The keys are as follows:

access The replacement text corresponding to the `name` key.

textaccess The replacement text corresponding to the `text` key.

firstaccess The replacement text corresponding to the `first` key.

pluralaccess The replacement text corresponding to the `plural` key.

firstpluralaccess The replacement text corresponding to the `firstplural` key.

symbolaccess The replacement text corresponding to the `symbol` key.

symbolpluralaccess The replacement text corresponding to the `symbolplural` key.

descriptionaccess The replacement text corresponding to the `description` key.

descriptionpluralaccess The replacement text corresponding to the `description-plural` key.

For example:

```
\newglossaryentry{tex}{name={\TeX},description={Document preparation language},access={TeX}}
```

Now `\gls{tex}` will be equivalent to

```
\BeginAccSupp{ActualText=TeX}\TeX\EndAccSupp{}
```

See Section 7 for further details. It is recommended that you also read the `accsupp` documentation.

3 Mfirstuc Package

The `glossaries` bundle is supplied with the package `mfirstuc` which provides the command:

`\makefirstuc`

```
\makefirstuc{<stuff>}
```

which makes the first object of `<stuff>` uppercase unless `<stuff>` starts with a control sequence followed by a non-empty group, in which case the first object in the group is converted to uppercase. Examples:

- `\makefirstuc{abc}` produces `Abc`

- `\makefirstuc{\emph{abc}}` produces *Abc* (`\MakeUppercase` has been applied to the letter “a” rather than `\emph`.) Note however that `\makefirstuc{\em abc}` produces *ABC* and `{\makefirstuc{\em abc}}` produces *abc*.
- `\makefirstuc{{\a}bc}` produces *Ábc*
- `\makefirstuc{\ae bc}` produces *Æbc*
- `\makefirstuc{{\ae}bc}` produces *Æbc*
- `\makefirstuc{{\ä}bc}` produces *Äbc*

Note that non-Latin or accented characters appearing at the start of the text must be placed in a group (even if you are using the `inputenc` package) due to expansion issues.

In version 1.02 of `mfirstuc`, a bug fix resulted in a change in output if the first object is a control sequence followed by an empty group. Prior to version 1.02, `\makefirstuc{\ae{}}bc` produced *æBc*. However as from version 1.02, it now produces *Æbc*.

Note also that

```
\newcommand{\abc}{abc}
\makefirstuc{\abc}
```

produces: *ABC*. This is because the first object in the argument of `\makefirstuc` is `\abc`, so it does `\MakeUppercase\abc`. Whereas:

```
\newcommand{\abc}{abc}
\expandafter\makefirstuc\expandafter{\abc}
```

produces: *Abc*. There is a short cut command which will do this:

`\xmakefirstuc`

```
\xmakefirstuc{\stuff}
```

This is equivalent to `\expandafter\makefirstuc\expandafter{\langle stuff \rangle}`. So

```
\newcommand{\abc}{abc}
\xmakefirstuc{\abc}
```

produces: *Abc*.

If you want to use an alternative command to convert to uppercase, for example `\MakeTextUppercase`,¹⁹ you can redefine the internal command `\@gls@makefirstuc`. For example:

```
\renewcommand{\@gls@makefirstuc}[1]{\MakeTextUppercase #1}
```

(Remember that command names that contain the `@` character must either be placed in packages or be placed between `\makeatletter` and `\makeatother`.)

¹⁹defined in the `textcase` package

4 Glossaries Documented Code

4.1 Package Definition

This package requires L^AT_EX 2_ε.

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{glossaries}[2010/07/10 v2.07 (NLCT)]
```

Required packages:

```
3 \RequirePackage{ifthen}
4 \RequirePackage{xkeyval}[2006/11/18]
5 \RequirePackage{mfirstuc}
6 \RequirePackage{xfor}
```

Need to use `\new@ifnextchar` instead of `\@ifnextchar` in commands that have a final optional argument (such as `\gls`) so require `.` Thanks to Morten Høgholm for suggesting this. (This has replaced using the `xspace` package.)

```
7 \RequirePackage{amsgen}
```

4.2 Package Options

toc The `toc` package option will add the glossaries to the table of contents. This is a boolean key, if the value is omitted it is taken to be true.

```
8 \define@boolkey{glossaries.sty}[gls]{toc}[true]{}%
```

numberline The `numberline` package option adds `\numberline` to `\addcontentsline`. Note that this option only has an effect if used in with `toc=true`.

```
9 \define@boolkey{glossaries.sty}[gls]{numberline}[true]{}%
```

The sectional unit used to start the glossary is stored in `\@@glossarysec`. If chapters are defined, this is initialised to `chapter`, otherwise it is initialised to `section`.

`\@@glossarysec`

```
10 \ifundefined{chapter}{\newcommand*{\@@glossarysec}{section}}{%
11 \newcommand*{\@@glossarysec}{chapter}}
```

section The `section` key can be used to set the sectional unit. If no unit is specified, use `section` as the default. The starred form of the named sectional unit will be used. If you want some other way to start the glossary section (e.g. a numbered section) you will have to redefine `\glossarysection`.

```
12 \define@choicekey{glossaries.sty}{section}{part,chapter,section,%
13 subsection,subsubsection,paragraph,subparagraph}[section]{}%
14 \renewcommand*{\@@glossarysec}{#1}}
```

Determine whether or not to use numbered sections.

`\@@glossarysecstar`

```
15 \newcommand*{\@@glossarysecstar}{*}
```

`\@glossaryseclabel`

```
16 \newcommand*{\@glossaryseclabel}{}%
```

`\glsautoprefix` Prefix to add before label if automatically generated:

```
17 \newcommand*{\glsautoprefix}{}%
```

`numberedsection`

```
18 \define@choicekey{glossaries.sty}{numberedsection}[\val\nr]{%
19 false,nolabel,autolabel}[nolabel]{%
20   \ifcase\nr\relax
21     \renewcommand*{\@glossarysecstar}{*}%
22     \renewcommand*{\@glossaryseclabel}{}%
23   \or
24     \renewcommand*{\@glossarysecstar}{}%
25     \renewcommand*{\@glossaryseclabel}{}%
26   \or
27     \renewcommand*{\@glossarysecstar}{}%
28     \renewcommand*{\@glossaryseclabel}{}%
29     \label{\glsautoprefix\@glo@type}%
30   \fi
31 }
```

The default glossary style is stored in `\@glossary@default@style`. This is initialised to `list`. (The `list` style is defined in the accompanying package described in [subsection 4.18](#).)

`\@glossary@default@style`

```
32 \newcommand*{\@glossary@default@style}{list}
```

`style` The default glossary style can be changed using the `style` package option. The value can be the name of any defined glossary style. The glossary style is set at the beginning of the document, so you can still use the `style` key to set a style that is defined in another package. This package comes with some predefined styles that are defined in [subsection 4.18](#).

```
33 \define@key{glossaries.sty}{style}{%
34 \renewcommand*{\@glossary@default@style}{#1}}
```

Each entry within a given glossary will have an associated number list. By default, this refers to the page numbers on which that entry has been used, but it can also refer to any counter used in the document (such as the section or equation counters). The default number list format displays the number list “as is”:

`\glossaryentrynumbers`

```
35 \newcommand*{\glossaryentrynumbers}[1]{#1}
```

`nonumberlist` Note that the entire number list for a given entry will be passed to `\glossaryentrynumbers` so any font changes will also be applied to the delimiters. The `nonumberlist` package option suppresses the number lists (this simply redefines `\glossaryentrynumbers` to ignore its argument).

```

36 \DeclareOptionX{nonumberlist}{%
37 \renewcommand*{\glossaryentrynumbers}[1]{}}

\@gls@loadlong
38 \newcommand*{\@gls@loadlong}{\RequirePackage{glossary-long}}

nolong This option prevents from being loaded. This means that the glossary styles that
use the longtable environment will not be available. This option is provided to
reduce overhead caused by loading unrequired packages.
39 \DeclareOptionX{nolong}{\renewcommand*{\@gls@loadlong}{}}

\@gls@loadsuper The package isn't loaded if isn't installed.
40 \IfFileExists{supertabular.sty}{%
41 \newcommand*{\@gls@loadsuper}{\RequirePackage{glossary-super}}}{%
42 \newcommand*{\@gls@loadsuper}{}}

nosuper This option prevents from being loaded. This means that the glossary styles that
use the supertabular environment will not be available. This option is provided to
reduce overhead caused by loading unrequired packages.
43 \DeclareOptionX{nosuper}{\renewcommand*{\@gls@loadsuper}{}}

\@gls@loadlist
44 \newcommand*{\@gls@loadlist}{\RequirePackage{glossary-list}}

nolist This option prevents from being loaded (to reduce overheads if required). Natu-
rally, the styles defined in will not be available if this option is used.
45 \DeclareOptionX{nolist}{\renewcommand*{\@gls@loadlist}{}}

\@gls@loadtree
46 \newcommand*{\@gls@loadtree}{\RequirePackage{glossary-tree}}

notree This option prevents from being loaded (to reduce overheads if required). Natu-
rally, the styles defined in will not be available if this option is used.
47 \DeclareOptionX{notree}{\renewcommand*{\@gls@loadtree}{}}

nostyles Provide an option to suppress all the predefined styles (in the event that the user
has custom styles that are not dependent on the predefined styles).
48 \DeclareOptionX{nostyles}{%
49 \renewcommand*{\@gls@loadlong}{}%
50 \renewcommand*{\@gls@loadsuper}{}%
51 \renewcommand*{\@gls@loadlist}{}%
52 \renewcommand*{\@gls@loadtree}{}%
53 \let\@glossary@default@style\relax
54 }

\glsdefmain Define the main glossary. This will be the first glossary to be displayed when using
\printglossaries.
55 \newcommand*{\glsdefmain}{%
56 \newglossary{main}{gls}{glo}{\glossaryname}%
57 }

```

Keep track of the default glossary. This is initialised to the main glossary, but can be changed if for some reason you want to make a secondary glossary the main glossary. This affects any commands that can optionally take a glossary name as an argument (or as the value of the `type` key in a key-value list). This was mainly done so that `\loadglsentries` can temporarily change `\glsdefaulttype` while it loads a file containing new glossary entries (see [subsection 4.9](#)).

`\glsdefaulttype`

```
58 \newcommand*{\glsdefaulttype}{main}
```

Keep track of which glossary the acronyms are in. This is initialised to `\glsdefaulttype`, but is changed by the `acronym` package option.

`\acronymtype`

```
59 \newcommand*{\acronymtype}{\glsdefaulttype}
```

The `nomain` option suppress the creation of the main glossary.

```
60 \DeclareOptionX{nomain}{%
61   \let\glsdefaulttype\relax
62   \renewcommand*{\glsdefmain}{}%
63 }
```

`acronym` The `acronym` option sets an associated conditional which is used in [subsection 4.16](#) to determine whether or not to define a separate glossary for acronyms.

```
64 \define@boolkey{glossaries.sty}[gls]{acronym}[true]{%
65   \DeclareAcronymList{acronym}%
66 }
```

`\@glsacronymlists` Comma-separated list of glossary labels indicating which glossaries contain acronyms. Note that `\SetAcronymStyle` must be used after adding labels to this macro.

```
67 \newcommand*{\@glsacronymlists}{}
```

`\@addtoacronymlists`

```
68 \newcommand*{\@addtoacronymlists}[1]{%
69   \ifx\@glsacronymlists\@empty
70     \protected@xdef\@glsacronymlists{#1}%
71   \else
72     \protected@xdef\@glsacronymlists{\@glsacronymlists,#1}%
73   \fi
74 }
```

`\DeclareAcronymList` Identifies the named glossary as a list of acronyms and adds to the list. (Doesn't check if the glossary exists, but checks if label already in list. Use `\SetAcronymStyle` after identifying all the acronym lists.)

```
75 \newcommand*{\DeclareAcronymList}[1]{%
76   \glsIfListOfAcronyms{#1}{\@addtoacronymlists{#1}}%
77 }
```

`\glsIfListOfAcronyms` `\glsIfListOfAcronyms{<label>}{<true part>}{<false part>}`

Determines if the glossary with the given label has been identified as being a list of acronyms.

```
78 \newcommand{\glsIfListOfAcronyms}[1]{%
79   \edef\@do@glis@islistofacronyms{%
80     \noexpand\@glis@islistofacronyms{#1}{\@glsacronymlists}}%
81   \@do@glis@islistofacronyms
82 }
```

Internal command requires label and list to be expanded:

```
83 \newcommand{\@glis@islistofacronyms}[4]{%
84   \def\@glis@islistofacronyms##1,#1,##2\end@glis@islistofacronyms{%
85     \def\@before{##1}\def\@after{##2}}%
86   \@glis@islistofacronyms,#2,#1,\@nil\end@glis@islistofacronyms
87   \ifx\@after\@nnil
```

Not found

```
88   #4%
89   \else
```

Found

```
90   #3%
91   \fi
92 }
```

`\if@glis@acronymlist` Convenient boolean.

```
93 \newif\if@glis@acronymlist
```

`\gls@check@is@acronymlist` Sets the above boolean if argument is a label representing a list of acronyms.

```
94 \newcommand*{\gls@check@is@acronymlist}[1]{%
95   \glsIfListOfAcronyms{#1}%
96   {\@glis@acronymlisttrue}{\@glis@acronymlistfalse}%
97 }
```

`\SetAcronymLists` Sets the “list of acronyms” list. Argument must be a comma-separated list of glossary labels. (Doesn’t check at this point if the glossaries exists.)

```
98 \newcommand*{\SetAcronymLists}[1]{%
99   \renewcommand*{\@glsacronymlists}{#1}%
100 }
```

`acronymlists`

```
101 \define@key{glossaries.sty}{acronymlists}{%
102   \@addtoacronymlists{#1}%
103 }
```

The default counter associated with the numbers in the glossary is stored in `\glscounter`. This is initialised to the page counter. This is used as the default counter when a new glossary is defined, unless a different counter is specified in the optional argument to `\newglossary` (see [subsection 4.6](#)).

`\glscounter`

```
104 \newcommand{\glscounter}{page}
```

counter The counter option changes the default counter. (This just redefines `\glscounter`.)

```
105 \define@key{glossaries.sty}{counter}{%
106   \renewcommand*{\glscounter}{#1}%
107 }
```

The glossary keys whose values are written to another file (i.e. `sort`, `name`, `description` and `symbol`) need to be sanitized, otherwise fragile commands would not be able to be used in `\newglossaryentry`. However, strange results will occur if you then use those fields in the document. As these fields are not normally used in the document, but are by default only used in the glossary, the default is to sanitize them. If however you want to use these values in the document (either by redefining commands like `\glsdisplay` or by using commands like `\glsentrydesc`) you will have to switch off the sanitization using the `sanitize` package option, but you will then have to use `\protect` to protect fragile commands when defining new glossary entries. The `sanitize` option takes a key-value list as its value, which can be used to switch individual values on and off. For example:

```
\usepackage[sanitize={description,name,symbol=false}]{glossaries}
```

will switch off the sanitization for the `symbol` key, but switch it on for the `description` and `name` keys. This would mean that you can use fragile commands in the `description` and `name` when defining a new glossary entry, but not for the `symbol`.

The default values are defined as:

`\@gls@sanitizedesc`

```
108 \newcommand*{\@gls@sanitizedesc}{\@onelevel@sanitize\@glo@desc}
```

`\@gls@sanitizename`

```
109 \newcommand*{\@gls@sanitizename}{\@onelevel@sanitize\@glo@name}
```

`\@gls@sanitizesymbol`

```
110 \newcommand*{\@gls@sanitizesymbol}{\@onelevel@sanitize\@glo@symbol}
```

(There is no equivalent for the `sort` key, since that is only provided for the benefit of `makeindex` or `xindy`, and so will always be sanitized.)

Before defining the `sanitize` package option, The key-value list for the `sanitize` value needs to be defined. These are all boolean keys. If they are not given a value, assume `true`.

Firstly the `description`. If set, it will redefine `\@gls@sanitizedesc` to use `\@onelevel@sanitize`, otherwise `\@gls@sanitizedesc` will do nothing.

```
111 \define@boolkey[gls]{sanitize}{description}[true]{%
112   \ifgls@sanitize@description
113     \renewcommand*{\@gls@sanitizedesc}{\@onelevel@sanitize\@glo@desc}%
114   \else
115     \renewcommand*{\@gls@sanitizedesc}{}%
116 \fi
117 }
```

Similarly for the name key:

```

118 \define@boolkey[glS]{sanitize}{name}[true]{%
119 \ifglS@sanitize@name
120   \renewcommand*{\@glS@sanitizename}{\@onelevel@sanitize\@glo@name}%
121 \else
122   \renewcommand*{\@glS@sanitizename}{}%
123 \fi}

```

and for the symbol key:

```

124 \define@boolkey[glS]{sanitize}{symbol}[true]{%
125 \ifglS@sanitize@symbol
126   \renewcommand*{\@glS@sanitizesymbol}{%
127   \@onelevel@sanitize\@glo@symbol}%
128 \else
129   \renewcommand*{\@glS@sanitizesymbol}{}%
130 \fi}

```

sanitize Now define the sanitize option. It can either take a key-val list as its value, or it can take the keyword none, which is equivalent to `description=false`, `symbol=false`, `name=false`:

```

131 \define@key{glossaries.sty}{sanitize}[description=true,symbol=true,
132 name=true]{%
133 \ifthenelse{\equal{#1}{none}}{%
134 \renewcommand*{\@glS@sanitizedesc}{}%
135 \renewcommand*{\@glS@sanitizename}{}%
136 \renewcommand*{\@glS@sanitizesymbol}{}%
137 }{\setkeys[glS]{sanitize}{#1}}%
138 }

```

translate Define translate option. If false don't set up multi-lingual support.

```

139 \define@boolkey{glossaries.sty}[glS]{translate}[true]{}

```

Set the default value:

```

140 \glstranslatefalse
141 \@ifpackageloaded{translator}{\glstranslatetrue}{%
142 \@ifpackageloaded{babel}{\glstranslatetrue}{%
143 \@ifpackageloaded{polyglossia}{\glstranslatetrue}{}}}

```

hyperfirst Set whether or not terms should have a hyperlink on first use.

```

144 \define@boolkey{glossaries.sty}[glS]{hyperfirst}[true]{}
145 \glshyperfirsttrue

```

footnote Set the long form of the acronym in footnote on first use.

```

146 \define@boolkey{glossaries.sty}[glSacr]{footnote}[true]{%
147 \ifthenelse{\boolean{glSacrdescription}}{%
148 {\renewcommand*{\@glS@sanitizedesc}{}%
149 }

```

description Allow acronyms to have a description (needs to be set using the `description` key in the optional argument of `\newacronym`).

```

150 \define@boolkey{glossaries.sty}[glsacr]{description}[true]{%
151   \renewcommand*{\@gls@sanitizesymbol}{}}%
152 }

```

smallcaps Define `\newacronym` to set the short form in small capitals.

```

153 \define@boolkey{glossaries.sty}[glsacr]{smallcaps}[true]{%
154   \renewcommand*{\@gls@sanitizesymbol}{}}%
155 }

```

smaller Define `\newacronym` to set the short form using `\smaller` which obviously needs to be defined by loading the appropriate package.

```

156 \define@boolkey{glossaries.sty}[glsacr]{smaller}[true]{%
157   \renewcommand*{\@gls@sanitizesymbol}{}}%
158 }

```

dua Define `\newacronym` to always use the long forms (i.e. don't use acronyms)

```

159 \define@boolkey{glossaries.sty}[glsacr]{dua}[true]{%
160   \renewcommand*{\@gls@sanitizesymbol}{}}%
161 }

```

shortcuts Define acronym shortcuts.

```

162 \define@boolkey{glossaries.sty}[glsacr]{shortcuts}[true]{}

```

\glsorder Stores the glossary ordering. This may either be “word” or “letter”. This passes the relevant information to `makeglossaries`. The default is word ordering.

```

163 \newcommand*{\glsorder}{word}

```

\@glsorder The ordering information is written to the auxiliary file for `makeglossaries`, so ignore the auxiliary information.

```

164 \newcommand*{\@glsorder}[1]{}

```

order

```

165 \define@choicekey{glossaries.sty}{order}{word,letter}{%
166   \def\glsorder{#1}}

```

\ifglxindy Provide boolean to determine whether `xindy` or `makeindex` will be used to sort the glossaries.

```

167 \newif\ifglxindy

```

The default is `makeindex`:

```

168 \glxindyfalse

```

Define package option to specify that `makeindex` will be used to sort the glossaries:

```

169 \DeclareOptionX{makeindex}{\glxindyfalse}

```


The xindy package option may have a value which in turn can be a key=value list. First define the keys for this sub-list. The boolean `glsnumbers` determines whether to automatically add the `glsnumbers` letter group.

```
170 \define@boolkey[gl]{xindy}{glsnumbers}[true]{}
171 \gl{xindy}{glsnumberstrue}
```

`\@xdy@main@language` Define what language to use for each glossary type (if a language is not defined for a particular glossary type the language specified for the main glossary is used.)

```
172 \def\@xdy@main@language{\rootlanguage}%
```

Define key to set the language

```
173 \define@key[gl]{xindy}{language}{\def\@xdy@main@language{#1}}
```

`\gls@codepage` Define the code page. If `\inputencodingname` is defined use that, otherwise have initialise with no codepage.

```
174 \@ifundefined{inputencodingname}{%
175   \def\gls@codepage{}}{%
176   \def\gls@codepage{\inputencodingname}
177 }
```

Define a key to set the code page.

```
178 \define@key[gl]{xindy}{codepage}{\def\gls@codepage{#1}}
```

Define package option to specify that xindy will be used to sort the glossaries:

```
179 \define@key{glossaries.sty}{xindy}[]{%
180   \gl{xindy}{true}
181   \setkeys[gl]{xindy}{#1}%
182 }
```

`\GlossariesWarning` Prints a warning message.

```
183 \newcommand*\GlossariesWarning[1]{%
184   \PackageWarning{glossaries}{#1}%
185 }
```

`\GlossariesWarningNoLine` Prints a warning message without the line number.

```
186 \newcommand*\GlossariesWarningNoLine[1]{%
187   \PackageWarningNoLine{glossaries}{#1}%
188 }
```

Define package option to suppress warnings

```
189 \DeclareOptionX{nowarn}{%
190   \renewcommand*\GlossariesWarning[1]{}%
191   \renewcommand*\GlossariesWarningNoLine[1]{}%
192 }
```

Process package options:

```
193 \ProcessOptionsX
```

If package is loaded, check to see if is installed, but only if translation is required.

```

194 \ifglstranslate
195 \ifpackageloaded{babel}{\IfFileExists{translator.sty}{%
196 \RequirePackage{translator}}{}}{}
197 \fi

```

If chapters are defined and the user has requested the section counter as a package option, `\@chapter` will be modified so that it adds a `section.<n>.0` target, otherwise entries placed before the first section of a chapter will have undefined links.

The same problem will also occur if a lower sectional unit is used, but this is less likely to happen. If it does, or if you change `\glscounter` to `section` later, you will have to specify a different counter for the entries that give rise to a `name{<section-level>.<n>.0}` non-existent warning (e.g. `\gls[counter=chapter]{label}`).

```

198 \ifthenelse{\equal{\glscounter}{section}}{%
199 \ifundefined{chapter}{}%
200 \let\@gls@old@chapter\@chapter
201 \def\@chapter[#1]#2{\@gls@old@chapter[#{#1}]{#2}%
202 \ifundefined{hyperdef}{\hyperdef{section}{\thesection}}{}}}%

```

`\@gls@onlypremakeg` Some commands only have an effect when used before `\makeglossaries`. So define a list of commands that should be disabled after `\makeglossaries`

```

203 \newcommand*{\@gls@onlypremakeg}{}

```

`\@onlypremakeg` Adds the specified control sequence to the list of commands that must be disabled after `\makeglossaries`.

```

204 \newcommand*{\@onlypremakeg}[1]{%
205 \ifx\@gls@onlypremakeg\@empty
206 \def\@gls@onlypremakeg{#1}%
207 \else
208 \expandafter\toks@\expandafter{\@gls@onlypremakeg}%
209 \edef\@gls@onlypremakeg{\the\toks@,\noexpand#1}%
210 \fi}

```

`\@disable@onlypremakeg` Disable all commands listed in `\@gls@onlypremakeg`

```

211 \newcommand*{\@disable@onlypremakeg}{%
212 \@for\@thiscs:=\@gls@onlypremakeg\do{%
213 \expandafter\@disable@premakecs\@thiscs%
214 }}

```

`\@disable@premakecs` Disables the given command.

```

215 \newcommand*{\@disable@premakecs}[1]{%
216 \def#1{\PackageError{glossaries}{\string#1\space may only be
217 used before \string\makeglossaries}{You can't use
218 \string#1\space after \string\makeglossaries}}%
219 }

```

4.3 Default values

This section sets up default values that are used by this package. Some of the names may already be defined (e.g. by) so `\providecommand` is used.

Main glossary title:

`\glossaryname`

```
220 \providecommand*\glossaryname{Glossary}
```

The title for the `acronym` glossary type (which is defined if `acronym` package option is used) is given by `\acronymname`. If the `acronym` package option is not used, `\acronymname` won't be used.

`\acronymname`

```
221 \providecommand*\acronymname{Acronyms}
```

`\glstocctitle` Sets the TOC title for the given glossary.

```
222 \newcommand*\glstocctitle{[1]{%
```

```
223 \def\glossarytocctitle{\csname @glotype@#1@title\endcsname}}
```

The following commands provide text for the headers used by some of the tabular-like glossary styles. Whether or not they get used in the glossary depends on the glossary style.

`\entryname`

```
224 \providecommand*\entryname{Notation}
```

`\descriptionname`

```
225 \providecommand*\descriptionname{Description}
```

`\symbolname`

```
226 \providecommand*\symbolname{Symbol}
```

`\pagelistname`

```
227 \providecommand*\pagelistname{Page List}
```

Labels for `makeindex`'s symbol and number groups:

`\glssymbolsgroupname`

```
228 \providecommand*\glssymbolsgroupname{Symbols}
```

`\glsnumpersgroupname`

```
229 \providecommand*\glsnumpersgroupname{Numbers}
```

`\glspluralsuffix` The default plural is formed by appending `\glspluralsuffix` to the singular form.

```
230 \newcommand*\glspluralsuffix{s}
```

`\seename`

```
231 \providecommand*\seename{see}
```

`\andname`

```
232 \providecommand*\andname{}\&}
```

Add multi-lingual support. Thanks to everyone who contributed to the translations from both comp.text.tex and via email.

`\addglossarytocaptions` If using `\glossaryname` should be defined in terms of `\translate`, but if `babel` is also loaded, it will redefine `\glossaryname` whenever the language is set, so override it. (Don't use `\addto` as doesn't define it.)

```
233 \newcommand*\addglossarytocaptions}[1]{%
234   \ifundefined{captions#1}{}%
235     \expandafter\let\expandafter\@gls@tmp\csname captions#1\endcsname
236     \expandafter\toks@\expandafter{\@gls@tmp
237       \renewcommand*\glossaryname{\translate{Glossary}}}%
238     }%
239     \expandafter\edef\csname captions#1\endcsname{\the\toks@}%
240   }%
241 }
```

```
242 \ifglstranslate
```

If is not install, used standard captions, otherwise load dictionary.

```
243 \ifpackageloaded{translator}{%
244   \usedictionary{glossaries-dictionary}%
245   \addglossarytocaptions{portuges}%
246   \addglossarytocaptions{portuguese}%
247   \addglossarytocaptions{brazil}%
248   \addglossarytocaptions{brazilian}%
249   \addglossarytocaptions{danish}%
250   \addglossarytocaptions{dutch}%
251   \addglossarytocaptions{afrikaans}%
252   \addglossarytocaptions{english}%
253   \addglossarytocaptions{UKenglish}%
254   \addglossarytocaptions{USenglish}%
255   \addglossarytocaptions{american}%
256   \addglossarytocaptions{australian}%
257   \addglossarytocaptions{british}%
258   \addglossarytocaptions{canadian}%
259   \addglossarytocaptions{newzealand}%
260   \addglossarytocaptions{french}%
261   \addglossarytocaptions{frenchb}%
262   \addglossarytocaptions{francais}%
263   \addglossarytocaptions{acadian}%
264   \addglossarytocaptions{canadien}%
265   \addglossarytocaptions{german}%
266   \addglossarytocaptions{germanb}%
267   \addglossarytocaptions{austrian}%
268   \addglossarytocaptions{naustrian}%
269   \addglossarytocaptions{ngerman}%
270   \addglossarytocaptions{irish}%

```

```

271 \addglossarytocaptions{italian}%
272 \addglossarytocaptions{magyar}%
273 \addglossarytocaptions{hungarian}%
274 \addglossarytocaptions{polish}%
275 \addglossarytocaptions{spanish}%
276 \renewcommand*{\glstttitle}[1]{%
277 \ifthenelse{\equal{#1}{main}}{%
278 \translatelet{\glossarytoctitle}{Glossary}}{%
279 \ifthenelse{\equal{#1}{acronym}}{%
280 \translatelet{\glossarytoctitle}{Acronyms}}{%
281 \def\glossarytoctitle{\csname @glotype@#1@title\endcsname}}}%
282 \renewcommand*{\glossaryname}{\translate{Glossary}}%
283 \renewcommand*{\acronymname}{\translate{Acronyms}}%
284 \renewcommand*{\entryname}{\translate{Notation (glossaries)}}%
285 \renewcommand*{\descriptionname}{%
286 \translate{Description (glossaries)}}%
287 \renewcommand*{\symbolname}{\translate{Symbol (glossaries)}}%
288 \renewcommand*{\pagelistname}{%
289 \translate{Page List (glossaries)}}%
290 \renewcommand*{\glssymbolsgroupname}{%
291 \translate{Symbols (glossaries)}}%
292 \renewcommand*{\glslnumbersgroupname}{%
293 \translate{Numbers (glossaries)}}%
294 }{%
295 \@ifpackageloaded{babel}%
296 {\RequirePackage{glossaries-babel}}%
297 {%
298 \@ifpackageloaded{polyglossia}{%
299 \RequirePackage{glossaries-polyglossia}}}%
300 }}
301 \fi

```

`\glspostdescription` The description terminator is given by `\glspostdescription` (except for the 3 and 4 column styles). This is a full stop by default:

```
302 \newcommand*{\glspostdescription}{.}
```

`\nopostdesc` Provide a means to suppress description terminator for a given entry. (Useful for entries with no description.) Has no effect outside the glossaries.

```
303 \newcommand*{\nopostdesc}{}

```

`\@nopostdesc` Suppress next description terminator.

```

304 \newcommand*{\@nopostdesc}{%
305 \let\org@glspostdescription\glspostdescription
306 \def\glspostdescription{%
307 \let\glspostdescription\org@glspostdescription}%
308 }

```

`\glspar` Provide means of having a paragraph break in glossary entries

```
309 \newcommand{\glspar}{\par}
```

`\setStyleFile` Sets the style file. The relevant extension is appended.

```
310 \ifglxindy
311   \newcommand{\setStyleFile}[1]{%
312     \renewcommand{\istfilename}{#1.xdy}}
313 \else
314   \newcommand{\setStyleFile}[1]{%
315     \renewcommand{\istfilename}{#1.ist}}
316 \fi
```

This command only has an effect prior to using `\makeglossaries`.

```
317 \@onlypremakeg\setStyleFile
```

The name of the `makeindex` or `xindy` style file is given by `\istfilename`. This file is created by `\writeist` (which is used by `\makeglossaries`) so redefining this command will only have an effect if it is done *before* `\makeglossaries`. As from v1.17, use `\setStyleFile` instead of directly redefining `\istfilename`.

`\istfilename`

```
318 \ifglxindy
319   \def\istfilename{jobname.xdy}
320 \else
321   \def\istfilename{jobname.ist}
322 \fi
```

The `makeglossaries` Perl script picks up this name from the auxiliary file. If the name ends with `.xdy` it calls `xindy` otherwise it calls `makeindex`. Since it's not required by \LaTeX , `\@istfilename` ignores its argument.

`\@istfilename`

```
323 \newcommand*{\@istfilename}[1]{}
```

This command is the value of the `page_compositor` `makeindex` key. Again, any redefinition of this command must take place *before* `\writeist` otherwise it will have no effect. As from 1.17, use `\glsSetCompositor` instead of directly redefining `\glscompositor`.

`\glscompositor`

```
324 \newcommand*{\glscompositor}{.}
```

`\glsSetCompositor` Sets the compositor.

```
325 \newcommand*{\glsSetCompositor}[1]{%
326   \renewcommand*{\glscompositor}{#1}}
```

Only use before `\makeglossaries`

```
327 \@onlypremakeg\glsSetCompositor
```

(The page compositor is usually defined as a dash when using `makeindex`, but most of the standard counters used by \LaTeX use a full stop as the compositor, which is why I have used it as the default.) If `xindy` is used `\glscompositor` only affects the `arabic-page-numbers` location class.

`\@glsAlphacompositor` This is only used by `xindy`. It specifies the compositor to use when location numbers are in the form $\langle letter \rangle \langle compositor \rangle \langle number \rangle$. For example, if `\@glsAlphacompositor` is set to “.” then it allows locations such as A.1 whereas if `\@glsAlphacompositor` is set to “-” then it allows locations such as A-1.

```
328 \newcommand*{\@glsAlphacompositor}{\glscompositor}
```

`\glsSetAlphaCompositor` Sets the alpha compositor.

```
329 \ifglsxindy
330   \newcommand*\glsSetAlphaCompositor[1]{%
331     \renewcommand*\@glsAlphacompositor{#1}}
332 \else
333   \newcommand*\glsSetAlphaCompositor[1]{%
334     \glsnxindywarning\glsSetAlphaCompositor}
335 \fi
```

Can only be used before `\makeglossaries`

```
336 \@onlypremakeg\glsSetAlphaCompositor
```

`\gls@suffiXF` Suffix to use for a two page list. This overrides the separator and the closing page number if set to something other than an empty macro.

```
337 \newcommand*{\gls@suffiXF}{}
```

`\glsSetSuffiXF` Sets the suffix to use for a two page list.

```
338 \newcommand*{\glsSetSuffiXF}[1]{%
339   \renewcommand*{\gls@suffiXF}{#1}}
```

Only has an effect when used before `\makeglossaries`

```
340 \@onlypremakeg\glsSetSuffiXF
```

`\gls@suffiFF` Suffix to use for a three page list. This overrides the separator and the closing page number if set to something other than an empty macro.

```
341 \newcommand*{\gls@suffiFF}{}
```

`\glsSetSuffiFF` Sets the suffix to use for a three page list.

```
342 \newcommand*{\glsSetSuffiFF}[1]{%
343   \renewcommand*{\gls@suffiFF}{#1}}
```

The command `\glsnumberformat` indicates the default format for the page numbers in the glossary. (Note that this is not the same as `\glossaryentrynumbers`, but applies to individual numbers or groups of numbers within an entry’s associated number list.) If hyperlinks are defined, it will use `\glshypernumber`, otherwise it will simply display its argument “as is”.

`\glsnumberformat`

```
344 \@ifundefined{hyperlink}{%
345   \newcommand*\glsnumberformat[1]{#1}}{%
346   \newcommand*\glsnumberformat[1]{\glshypernumber{#1}}}
```

Individual numbers in an entry’s associated number list are delimited using `\delimN` (which corresponds to the `delim_n makeindex` keyword). The default value is a comma followed by a space.

`\delimN`

```
347 \newcommand{\delimN}{, }
```

A range of numbers within an entry’s associated number list is delimited using `\delimR` (which corresponds to the `delim_r makeindex` keyword). The default is an en-dash.

`\delimR`

```
348 \newcommand{\delimR}{--}
```

The glossary preamble is given by `\glossarypreamble`. This will appear after the glossary sectioning command, and before the `theglossary` environment. It is designed to allow the user to add information pertaining to the glossary (e.g. “page numbers in italic indicate the primary definition”) therefore `\glossarypreamble` shouldn’t be affected by the glossary style. (So if you define your own glossary style, don’t have it change `\glossarypreamble`.) The preamble is empty by default. If you have multiple glossaries, and you want a different preamble for each glossary, you will need to use `\printglossary` for each glossary type, instead of `\printglossaries`, and redefine `\glossarypreamble` before each `\printglossary`.

`\glossarypreamble`

```
349 \newcommand*{\glossarypreamble}{}%
```

The glossary postamble is given by `\glossarypostamble`. This is provided to allow the user to add something after the end of the `theglossary` environment (again, this shouldn’t be affected by the glossary style). It is, of course, possible to simply add the text after `\printglossary`, but if you only want the postamble to appear after the first glossary, but not after subsequent glossaries, you can do something like:

```
\renewcommand{\glossarypostamble}{For a complete list of terms
see \cite{blah}\gdef\glossarypreamble{}}
```

`\glossarypostamble`

```
350 \newcommand*{\glossarypostamble}{}%
```

The sectioning command that starts a glossary is given by `\glossarysection`. (This does not form part of the glossary style, and so should not be changed by a glossary style.) If `\phantomsection` is defined, it uses `\p@glossarysection`, otherwise it uses `\@glossarysection`.

`\glossarysection`

```
351 \newcommand*{\glossarysection}[2][\@gls@title]{%
352   \def\@gls@title{#2}%
\phantomsection
\section{#1}%
\gls@title{#2}%
}
```



```

353 \@ifundefined{phantomsection}{%
354 \@glossarysection{#1}{#2}}{\p@glossarysection{#1}{#2}}%
355 \glossarymark{\glossarytoctitle}%
356 }

```

`\glossarymark` Sets the header mark for the glossary. Takes the glossary short (TOC) title as the argument.

```

357 \@ifundefined{glossarymark}{%
358 \newcommand{\glossarymark}[1]{\@mkboth{#1}{#1}}
359 }{%
360 \GlossariesWarning{overriding \string\glossarymark}%
361 \ifclassloaded{memoir}%
362 {
363 \renewcommand{\glossarymark}[1]{%
364 \markboth{\memUHead{#1}}{\memUHead{#1}}%
365 }
366 }
367 {
368 \renewcommand{\glossarymark}[1]{\@mkboth{#1}{#1}}
369 }
370 }

```

The required sectional unit is given by `\@glossarysec` which was defined by the `section` package option. The starred form of the command is chosen. If you don't want any sectional command, you will need to redefine `\glossarysection`. The sectional unit can be changed, if different sectional units are required.

`\setglossarysection`

```

371 \newcommand*\setglossarysection[1]{%
372 \setkeys{glossaries.sty}{section=#1}}

```

The command `\@glossarysection` indicates how to start the glossary section if `\phantomsection` is not defined.

`\@glossarysection`

```

373 \newcommand*\@glossarysection[2]{%
374 \ifx\@glossarysecstar\@empty
375 \csname\@glossarysec\endcsname{#2}%
376 \else
377 \csname\@glossarysec\endcsname*{#2}%
378 \@gls@toc{#1}{\@glossarysec}%
379 \fi
380 \@glossaryseclabel}

```

As `\@glossarysection`, but put in `\phantomsection`, and swap where `\@gls@toc` goes. If using chapters do a `\clearpage`. This ensures that the hyper link from the table of contents leads to the line above the heading, rather than the line below it.

`\p@glossarysection`

```
381 \newcommand*{\p@glossarysection}[2]{%
382 \glsclearpage
383 \phantomsection
384 \ifx\@glossarysecstar\@empty
385   \csname\@glossarysec\endcsname{#2}%
386 \else
387   \@gls@toc{#1}{\@glossarysec}%
388   \csname\@glossarysec\endcsname*{#2}%
389 \fi
390 \@glossaryseclabel}
```

The `\gls@doclearpage` command is used to issue a `\clearpage` (or `\cleardoublepage`) depending on whether the glossary sectional unit is a chapter. If the sectional unit is something else, do nothing.

`\gls@doclearpage`

```
391 \newcommand*{\gls@doclearpage}{%
392 \ifthenelse{\equal{\@glossarysec}{chapter}}{%
393 \@ifundefined{cleardoublepage}{\clearpage}{\cleardoublepage}}{}%
394 }
```

`\glsclearpage` This just calls `\gls@doclearpage`, but it makes it easier to have a user command so that the user can override it.

```
395 \newcommand*{\glsclearpage}{\gls@doclearpage}
```

The glossary is added to the table of contents if `glstoc` flag set. If it is set, `\@gls@toc` will add a line to the `.toc` file, otherwise it will do nothing. (The first argument to `\@gls@toc` is the title for the table of contents, the second argument is the sectioning type.)

`\@gls@toc`

```
396 \newcommand*{\@gls@toc}[2]{%
397 \ifglstoc
398   \ifglslnumberline
399     \addcontentsline{toc}{#2}{\numberline{#1}%
400   \else
401     \addcontentsline{toc}{#2}{#1}%
402   \fi
403 \fi}
```

4.4 Xindy

This section defines commands that only have an effect if `xindy` is used to sort the glossaries.

`\glsnoxindywarning` Issues a warning if `xindy` hasn't been specified. These warnings can be suppressed by redefining `\glsnoxindywarning` to ignore its argument

```
404 \newcommand*{\glsnoxindywarning}[1]{%
```

```

405 \GlossariesWarning{Not in xindy mode --- ignoring \string#1}%
406 }

\@xdyattributes Define list of attributes (\string is used in case the double quote character has
                been made active)
407 \ifglxsindy
408 \edef\@xdyattributes{\string"default\string"}%
409 \fi

\@xdylocref Define list of markup location references.
410 \ifglxsindy
411 \def\@xdylocref{}
412 \fi

\GlsAddXdyAttribute Adds an attribute.
413 \ifglxsindy
414 \newcommand*\GlsAddXdyAttribute[1]{%
415 \edef\@xdyattributes{\@xdyattributes ^^J \string"#1\string"}%
416 \expandafter\toks@\expandafter{\@xdylocref}%
417 \edef\@xdylocref{\the\toks@ ^^J%
418 (markup-locref
419 :open \string"\string~n\string\setentrycounter
420 {\noexpand\glscounter}}%
421 \expandafter\string\csname#1\endcsname
422 \expandafter@gobble\string\{\string" ^^J
423 :close \string"\expandafter@gobble\string\}\string" ^^J
424 :attr \string"#1\string")}}

    Only has an effect before \writeist:
425 \@onlypremakeg\GlsAddXdyAttribute
426 \else
427 \newcommand*\GlsAddXdyAttribute[1]{%
428 \glsnosindywarning\GlsAddXdyAttribute}
429 \fi

    Add known attributes:
430 \ifglxsindy
431 \GlsAddXdyAttribute{glnumberformat}
432 \GlsAddXdyAttribute{textrm}
433 \GlsAddXdyAttribute{textsf}
434 \GlsAddXdyAttribute{texttt}
435 \GlsAddXdyAttribute{textbf}
436 \GlsAddXdyAttribute{textmd}
437 \GlsAddXdyAttribute{textit}
438 \GlsAddXdyAttribute{textup}
439 \GlsAddXdyAttribute{textsl}
440 \GlsAddXdyAttribute{textsc}
441 \GlsAddXdyAttribute{emph}
442 \GlsAddXdyAttribute{glshypernumber}
443 \GlsAddXdyAttribute{hyperrrm}

```

```

444 \GlsAddXdyAttribute{hypersf}
445 \GlsAddXdyAttribute{hypertt}
446 \GlsAddXdyAttribute{hyperbf}
447 \GlsAddXdyAttribute{hypermd}
448 \GlsAddXdyAttribute{hyperit}
449 \GlsAddXdyAttribute{hyperup}
450 \GlsAddXdyAttribute{hypersl}
451 \GlsAddXdyAttribute{hypersc}
452 \GlsAddXdyAttribute{hyperemph}
453 \fi

\@xdyuseralphabets List of additional alphabets
454 \def\@xdyuseralphabets{}

\GlsAddXdyAlphabet \GlsAddXdyAlphabet{<name>}{<definition>} adds a new alphabet called <name>.
The definition must use xindy syntax.
455 \ifglsxindy
456 \newcommand*{\GlsAddXdyAlphabet}[2]{%
457 \edef\@xdyuseralphabets{%
458 \@xdyuseralphabets ^^J
459 (define-alphabet "#1" (#2))}%
460 \else
461 \newcommand*{\GlsAddXdyAlphabet}[2]{%
462 \glsnxindywarning\GlsAddXdyAlphabet}
463 \fi

\@xdyuserlocationdefs List of additional location definitions (separated by ^^J)
464 \def\@xdyuserlocationdefs{}

\@xdyuserlocationnames List of additional user location names
465 \def\@xdyuserlocationnames{}

\GlsAddXdyLocation \GlsAddXdyLocation{<name>}{<definition>} Define a new location called <name>.
The definition must use xindy syntax. (Note that this doesn't check to see if the
location is already defined. That is left to xindy to complain about.)
466 \ifglsxindy
467 \newcommand*{\GlsAddXdyLocation}[2]{%
468 \edef\@xdyuserlocationdefs{%
469 \@xdyuserlocationdefs ^^J%
470 (define-location-class \string"#1\string"^^J\space\space
471 \space(#2))
472 }%
473 \edef\@xdyuserlocationnames{%
474 \@xdyuserlocationnames^^J\space\space\space
475 \string"#1\string"}%
476 }

Only has an effect before \writeist:
477 \@onlypremakeg\GlsAddXdyLocation

```

```

478 \else
479   \newcommand*\GlsAddXdyLocation[2]{%
480     \glsnoxywarning\GlsAddXdyLocation}
481 \fi

```

`\@xdylocationclassorder` Define location class order

```

482 \ifglxindy
483   \edef\@xdylocationclassorder{^^J\space\space\space
484     \string"roman-page-numbers\string"^^J\space\space\space
485     \string"arabic-page-numbers\string"^^J\space\space\space
486     \string"arabic-section-numbers\string"^^J\space\space\space
487     \string"alpha-page-numbers\string"^^J\space\space\space
488     \string"Roman-page-numbers\string"^^J\space\space\space
489     \string"Alpha-page-numbers\string"^^J\space\space\space
490     \string"Appendix-page-numbers\string"
491     \@xdyuserlocationnames^^J\space\space\space
492     \string"see\string"
493   }
494 \fi

```

Change the location order.

`\GlsSetXdyLocationClassOrder`

```

495 \ifglxindy
496   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
497     \def\@xdylocationclassorder{#1}}
498 \else
499   \newcommand*\GlsSetXdyLocationClassOrder[1]{%
500     \glsnoxywarning\GlsSetXdyLocationClassOrder}
501 \fi

```

`\@xdysortrules` Define sort rules

```

502 \ifglxindy
503   \def\@xdysortrules{}
504 \fi

```

`\GlsAddSortRule` Add a sort rule

```

505 \ifglxindy
506   \newcommand*\GlsAddSortRule[2]{%
507     \expandafter\toks@\expandafter{\@xdysortrules}%
508     \protected@edef\@xdysortrules{\the\toks@ ^^J
509       (sort-rule \string"#1\string" \string"#2\string"))}%
510   }
511 \else
512   \newcommand*\GlsAddSortRule[2]{%
513     \glsnoxywarning\GlsAddSortRule}
514 \fi

```

`\@xdyrequiredstyles` Define list of required styles (this should be a comma-separated list of xindy styles)

```

515 \ifglxindy
516   \def\@xdyrequiredstyles{tex}
517 \fi

\GlsAddXdyStyle  Add a xindy style to the list of required styles
518 \ifglxindy
519   \newcommand*\GlsAddXdyStyle[1]{%
520     \edef\@xdyrequiredstyles{\@xdyrequiredstyles,#1}}%
521 \else
522   \newcommand*\GlsAddXdyStyle[1]{%
523     \glsnxindywarning\GlsAddXdyStyle}
524 \fi

\GlsSetXdyStyles  Reset the list of required styles
525 \ifglxindy
526   \newcommand*\GlsSetXdyStyles[1]{%
527     \edef\@xdyrequiredstyles{#1}}
528 \else
529   \newcommand*\GlsSetXdyStyles[1]{%
530     \glsnxindywarning\GlsSetXdyStyles}
531 \fi

\findrootlanguage  The root language name is required by xindy. This information is for makeglossaries
                    to pass to xindy. Since \language name only stores the regional dialect rather than
                    the root language name, some trickery is required to determine the root language.
532 \ifglxindy
533   \@ifpackageloaded{babel}{%
                    Need to parse babel.sty to determine the root language. This code was provided
                    by Enrico Gregorio.
534   \def\findrootlanguage{\begingroup
535     \escapechar=-1\relax
                    normalize \language name to category 12 chars
536     \edef\language name{%
537       \expandafter\string\csname\language name\endcsname}%
                    disable babel.sty useless commands
538     \def\NeedsTeXFormat##1[##2]{}%
539     \def\ProvidesPackage##1[##2]{}%
540     \let\LdfInit\relax
541     \def\languageattribute##1##2{%
                    change the meaning of \DeclareOption
542     \def\DeclareOption##1##2{%
                    at \DeclareOption* we end
543       \ifx##1*\expandafter\endinput\else
                    else we build a string with the first argument
544       \edef\testlanguage{\expandafter\string\csname##1\endcsname}%

```

if `\testlanguage` and `\language` are the same we execute the second argument

```

545     \ifx\testlanguage\language##2\fi
546   \fi}

  almost all options of babel are \input{<name>.ldf}
547   \def\input##1{\stripldf##1}%

  we put the root language name in \rootlanguage
548   \def\stripldf##1.ldf{\gdef\rootlanguage{##1}}%

  now input babel.sty, using the primitive \input
549   \@@input babel.sty
550   \endgroup}%
551 }{%

  hasn't been loaded, so check if has been loaded
552   \ifpackageloaded{ngerman}{%
553     \def\findrootlanguage{%
554       \def\rootlanguage{german}}}%
555   }{%

  Neither babel nor ngerman have been loaded, so assume the root language is English
556     \def\findrootlanguage{%
557       \def\rootlanguage{english}}}%
558   }%
559 }%
560 \fi

```

`\rootlanguage` Set default root language to English.

```

561 \def\rootlanguage{english}

```

`\xdylanguage` The xindy language setting is required by `makeglossaries`, so provide a command for `makeglossaries` to pick up the information from the auxiliary file. This command is not needed by the `glossaries` package, so define it to ignore its arguments.

```

562 \def\xdylanguage#1#2{}

```

`\GlsSetXdyLanguage` Define a command that allows the user to set the language for a given glossary type. The first argument indicates the glossary type. If omitted the main glossary is assumed.

```

563 \ifglxindy
564   \newcommand*\GlsSetXdyLanguage[2][\glsdefaulttype]{%
565     \ifglossaryexists{#1}{%
566       \expandafter\def\csname @xdy@#1@language\endcsname{#2}%
567     }{%
568       \PackageError{glossaries}{Can't set language type for
569         glossary type '#1' --- no such glossary}{%
570         You have specified a glossary type that doesn't exist}}
571   \else

```

```

572 \newcommand*\GlsSetXdyLanguage[2][]{%
573   \glsnoxywarning\GlsSetXdyLanguage}
574 \fi

```

`\@gls@codepage` The xindy codepage setting is required by `makeglossaries`, so provide a command for `makeglossaries` to pick up the information from the auxiliary file. This command is not needed by the `glossaries` package, so define it to ignore its arguments.

```

575 \def\@gls@codepage#1#2{}

```

`\GlsSetXdyCodePage` Define command to set the code page.

```

576 \ifglsxindy
577   \newcommand*\GlsSetXdyCodePage[1]{%
578     \renewcommand*\gls@codepage{#1}%
579   }
580 \else
581   \newcommand*\GlsSetXdyCodePage[1]{%
582     \glsnoxywarning\GlsSetXdyCodePage}
583 \fi

```

`\@xdylettergroups` Store letter group definitions.

```

584 \ifglsxindy
585   \ifglsxindy@glsnumbers
586     \def\@xdylettergroups{(define-letter-group
587       \string"glsnumbers\string"^^J\space\space\space
588       :prefixes (\string"0\string" \string"1\string"
589       \string"2\string" \string"3\string" \string"4\string"
590       \string"5\string" \string"6\string" \string"7\string"
591       \string"8\string" \string"9\string")^^J\space\space\space
592       :before \string"\@glsfirstletter\string")}
593   \else
594     \def\@xdylettergroups{}
595   \fi
596 \fi
597 % \end{macrocode}
598 %\end{macro}
599 %
600 %\begin{macro}{\GlsAddLetterGroup}
601 % Add a new letter group. The first argument is the name
602 % of the letter group. The second argument is the \app{xindy}
603 % code specifying prefixes and ordering.
604 % \begin{macrocode}
605 \newcommand*\GlsAddLetterGroup[2]{%
606   \expandafter\toks@\expandafter{\@xdylettergroups}%
607   \protected@edef\@xdylettergroups{the\toks@^^J%
608   (define-letter-group \string"#1\string"^^J\space\space\space#2)}%
609 }%

```


4.5 Loops and conditionals

`\forall glossaries` To iterate through all glossaries (or comma-separated list of glossary names given in optional argument) use:

```
\forall glossaries[<glossary list>]{<cmd>}{<code>}
```

where *<cmd>* is a control sequence which will be set to the name of the glossary in the current iteration.

```
610 \newcommand*\forallglossaries[3][\@glo@types]{%
611   \@for#2:=#1\do{\ifx#2\@empty\else#3\fi}%
612 }
```

`\forall glossentries` To iterate through all entries in a given glossary use:

```
\forall glossentries[<type>]{<cmd>}{<code>}
```

where *<type>* is the glossary label and *<cmd>* is a control sequence which will be set to the entry label in the current iteration.

```
613 \newcommand*\forallglossentries[3][\glsdefaulttype]{%
614   \edef\@glo@list{\csname glo@list@#1\endcsname}%
615   \@for#2:=\@glo@list\do{\ifx#2\@empty\else#3\fi}%
616 }
```

`\forall glossentries` To iterate through all glossary entries over all glossaries listed in the optional argument (the default is all glossaries) use:

```
\forall glossentries[<glossary list>]{<cmd>}{<code>}
```

Within `\forall glossentries`, the current glossary type is given by `\@thisglo@`.

```
617 \newcommand*\forallglossentries[3][\@glo@types]{%
618   \expandafter\forallglossaries\expandafter[#1]{\@thisglo@}{%
619     \forallglossentries[\@thisglo@]{#2}{#3}}}
```

`\if glossaryexists` To check to see if a glossary exists use:

```
\if glossaryexists{<type>}{<true-text>}{<false-text>}
```

where *<type>* is the glossary's label.

```
620 \newcommand{\ifglossaryexists}[3]{%
621   \@ifundefined{glo@#1@out}{#3}{#2}%
622 }
```

`\if glossentryexists` To check to see if a glossary entry has been defined use:

```
\if glossentryexists{<label>}{<true text>}{<false text>}
```

where *<label>* is the entry's label.

```
623 \newcommand{\ifglossentryexists}[3]{%
624   \@ifundefined{glo@#1@name}{#3}{#2}}
```

`\ifglused` To determine if given glossary entry has been used in the document text yet use:

```
\ifglused{<label>}{<true text>}{<false text>}
```

where `<label>` is the entry's label. If true it will do `<true text>` otherwise it will do `<false text>`.

```
625 \newcommand*{\ifglused}[3]{\ifthenelse{\boolean{glo@#1@flag}}{#2}{#3}}
```

The following two commands will cause an error if the given condition fails:

```
\glstoifexists \glstoifexists{<label>}{<code>}
```

Generate an error if entry specified by `<label>` doesn't exist, otherwise do `<code>`.

```
626 \newcommand{\glstoifexists}[2]{%
627   \ifglentryexists{#1}{#2}{%
628     \PackageError{glossaries}{Glossary entry '#1' has not been
629     defined}{You need to define a glossary entry before you
630     can use it.}}%
631 }
```

```
\glstoifnoexists \glstoifnoexists{<label>}{<code>}
```

The opposite: only do second argument if the entry doesn't exist. Generate an error message if it exists.

```
632 \newcommand{\glstoifnoexists}[2]{%
633   \ifglentryexists{#1}{%
634     \PackageError{glossaries}{Glossary entry '#1' has already
635     been defined}{}}{#2}%
636 }
```

4.6 Defining new glossaries

A comma-separated list of glossary names is stored in `\@glo@types`. When a new glossary type is created, its identifying name is added to this list. This is used by commands that iterate through all glossaries (such as `\makeglossaries` and `\printglossaries`).

```
\@glo@types
```

```
637 \newcommand*{\@glo@types}{,}
```

A new glossary type is defined using `\newglossary`. Syntax:

```
\newglossary[<log-ext>]{<name>}{<in-ext>}{<out-ext>}{<title>}[<counter>]
```

where `<log-ext>` is the extension of the `makeindex` transcript file, `<in-ext>` is the extension of the glossary input file (read in by `\printglossary` and created by `makeindex`), `<out-ext>` is the extension of the glossary output file which is read in by `makeindex` (lines are written to this file by the `\glossary` command), `<title>` is the title of the glossary that is used in `\glossarysection` and `<counter>` is the default

counter to be used by entries belonging to this glossary. The `makeglossaries` Perl script reads in the relevant extensions from the auxiliary file, and passes the appropriate file names and switches to `makeindex`.

`\newglossary`

```
638 \newcommand*{\newglossary}[5][glg]{%
639 \ifglossaryexists{#2}{%
640   \PackageError{glossaries}{Glossary type ‘#2’ already exists}{%
641     You can’t define a new glossary called ‘#2’ because it already
642     exists}%
643 }{%
```

Check if default has been set

```
644 \ifx\glsdefaulttype\relax
645   \gdef\glsdefaulttype{#2}%
646 \fi
```

Add this to the list of glossary types:

```
647 \toks@{#2}\edef\@glo@types{\@glo@types\the\toks@,}%
```

Define a comma-separated list of labels for this glossary type, so that all the entries for this glossary can be reset with a single command. When a new entry is created, its label is added to this list.

```
648 \expandafter\gdef\csname glolist@#2\endcsname{,}%
```

Store details of this new glossary type:

```
649 \expandafter\def\csname @glotype@#2@in\endcsname{#3}%
650 \expandafter\def\csname @glotype@#2@out\endcsname{#4}%
651 \expandafter\def\csname @glotype@#2@title\endcsname{#5}%
652 \protected@write\@auxout{\string\@newglossary{#2}{#1}{#3}{#4}}%
```

How to display this entry in the document text (uses `\glsdisplay` and `\glsdisplayfirst` by default). These can be redefined by the user later if required (see `\defglsdisplay` and `\defglsdisplayfirst`). These may already have been defined if this has been specified as a list of acronyms.

```
653 \@ifundefined{gls@#2@display}{%
654   \expandafter\gdef\csname gls@#2@display\endcsname{%
655     \glsdisplay}}{%
656 \ifundefined{gls@#2@displayfirst}{%
657   \expandafter\gdef\csname gls@#2@displayfirst\endcsname{%
658     \glsdisplayfirst}}{%
```

Find out if the final optional argument has been specified, and use it to set the counter associated with this glossary. (Uses `\glscounter` if no optional argument is present.)

```
659 \ifnextchar[{\@gls@setcounter{#2}}%
660   {\@gls@setcounter{#2}[\glscounter]}}}
```

`\altnewglossary`

```
661 \newcommand*{\altnewglossary}[3]{%
662   \newglossary[#2-glg]{#1}{#2-gls}{#2-glo}{#3}%
663 }
```

Only define new glossaries in the preamble:

```
664 \onlypreamble{\newglossary}
```

Only define new glossaries before `\makeglossaries`

```
665 \onlypremakeg\newglossary
```

`\newglossary` is used to specify the file extensions for the `makeindex` input, output and transcript files. It is written to the auxiliary file by `\newglossary`. Since it is not used by L^AT_EX, `\newglossary` simply ignores its arguments.

`\newglossary`

```
666 \newcommand*{\newglossary}[4]{}
```

Store counter to be used for given glossary type (the first argument is the glossary label, the second argument is the name of the counter):

`\gls@setcounter`

```
667 \def\gls@setcounter#1[#2]{%
```

```
668 \expandafter\def\csname @gls@#1@counter\endcsname{#2}%
```

```
669 }
```

Get counter associated with given glossary (the argument is the glossary label):

`\gls@getcounter`

```
670 \newcommand*{\gls@getcounter}[1]{%
```

```
671 \csname @gls@#1@counter\endcsname}
```

Define the main glossary. This will be the first glossary to be displayed when using `\printglossaries`.

```
672 \glsdefmain
```

4.7 Defining new entries

New glossary entries are defined using `\newglossaryentry`. This command requires a label and a key-value list that defines the relevant information for that entry. The definition for these keys follows. Note that the `name`, `description` and `symbol` keys will be sanitized later, depending on the value of the package option `sanitize` (this means that if some of the keys haven't been defined, they can be constructed from the `name` and `description` key before they are sanitized).

name The `name` key indicates the name of the term being defined. This is how the term will appear in the glossary. The `name` key is required when defining a new glossary entry.

```
673 \define@key{glossentry}{name}{%
```

```
674 \def\glo@name{#1}%
```

```
675 }
```

description The `description` key is usually only used in the glossary, but can be made to appear in the text by redefining `\glsdisplay` and `\glsdisplayfirst` (or using `\defglsdisplay` and `\defglsdisplayfirst`), however, you will have to disable

the `sanitize` option (using the `sanitize` package option, `sanitize={description=false}`, and protect fragile commands). The `description` key is required when defining a new glossary entry. (Be careful not to make the description too long, because `makeindex` has a limited buffer. `\glo@desc` is defined to be a short command to discourage lengthy descriptions for this reason. If you do have a very long description, or if you require paragraph breaks, define a separate command that contains the description, and use it as the value to the `description` key.)

```
676 \define@key{glossentry}{description}{%
677 \def\@glo@desc{#1}%
678 }
```

descriptionplural

```
679 \define@key{glossentry}{descriptionplural}{%
680 \def\@glo@descplural{#1}%
681 }
```

sort The `sort` key needs to be sanitized here (the `sort` key is provided for `makeindex`'s benefit, not for use in the document). The `sort` key is optional when defining a new glossary entry. If omitted, the value is given by *<name>* *<description>*.

```
682 \define@key{glossentry}{sort}{%
683 \def\@glo@sort{#1}}
```

text The `text` key determines how the term should appear when used in the document (i.e. outside of the glossary). If omitted, the value of the `name` key is used instead.

```
684 \define@key{glossentry}{text}{%
685 \def\@glo@text{#1}%
686 }
```

plural The `plural` key determines how the plural form of the term should be displayed in the document. If omitted, the plural is constructed by appending `\glspluralsuffix` to the value of the `text` key.

```
687 \define@key{glossentry}{plural}{%
688 \def\@glo@plural{#1}%
689 }
```

first The `first` key determines how the entry should be displayed in the document when it is first used. If omitted, it is taken to be the same as the value of the `text` key.

```
690 \define@key{glossentry}{first}{%
691 \def\@glo@first{#1}%
692 }
```

firstplural The `firstplural` key is used to set the plural form for first use, in the event that the plural is required the first time the term is used. If omitted, it is constructed by appending `\glspluralsuffix` to the value of the `first` key.

```
693 \define@key{glossentry}{firstplural}{%
694 \def\@glo@firstplural{#1}%
695 }
```

symbol The `symbol` key is ignored by most of the predefined glossary styles, and defaults to `\relax` if omitted. It is provided for glossary styles that require an associated symbol, as well as a name and description. To make this value appear in the glossary, you need to redefine `\glossaryentryfield` so that it uses its fourth parameter. If you want this value to appear in the text when the term is used by commands like `\gls`, you will need to change `\glsdisplay` and `\glsdisplayfirst` (either explicitly for all glossaries or via `\defglsdisplay` and `\defglsdisplayfirst` for individual glossaries).

```
696 \define@key{glossentry}{symbol}{%
697 \def\@glo@symbol{#1}%
698 }
```

symbolplural

```
699 \define@key{glossentry}{symbolplural}{%
700 \def\@glo@symbolplural{#1}%
701 }
```

type The `type` key specifies to which glossary this entry belongs. If omitted, the default glossary is used.

```
702 \define@key{glossentry}{type}{%
703 \def\@glo@type{#1}}
```

counter The `counter` key specifies the name of the counter associated with this glossary entry:

```
704 \define@key{glossentry}{counter}{%
705 \@ifundefined{c@#1}{\PackageError{glossaries}{There is no counter
706 called ‘#1’}{The counter key should have the name of a valid
707 counter as its value}}{%
708 \def\@glo@counter{#1}}}
```

see The `see` key specifies a list of cross-references

```
709 \define@key{glossentry}{see}{%
710 \def\@glo@see{#1}}
```

parent The `parent` key specifies the parent entry, if required.

```
711 \define@key{glossentry}{parent}{%
712 \def\@glo@parent{#1}}
```

nonumberlist The `nonumberlist` key suppresses the number list for the given entry.

```
713 \define@key{glossentry}{nonumberlist}[none]{%
714 \def\@glo@prefix{\glsnonextpages}}
```

Define some generic user keys. (6 ought to be enough!)

user1

```
715 \define@key{glossentry}{user1}{%
716 \def\@glo@useri{#1}%
717 }
```

```

user2
718 \define@key{glossentry}{user2}{%
719   \def\@glo@userii{#1}%
720 }

user3
721 \define@key{glossentry}{user3}{%
722   \def\@glo@useriii{#1}%
723 }

user4
724 \define@key{glossentry}{user4}{%
725   \def\@glo@useriv{#1}%
726 }

user5
727 \define@key{glossentry}{user5}{%
728   \def\@glo@userv{#1}%
729 }

user6
730 \define@key{glossentry}{user6}{%
731   \def\@glo@uservi{#1}%
732 }

\@glsnname Define command to generate error if name key is missing.
733 \newcommand*{\@glsnname}{%
734   \PackageError{glossaries}{name key required in
735     \string\newglossaryentry\space for entry ‘\@glo@label’}{You
736     haven’t specified the entry name}}

\@glsdefaultplural Define command to set default plural.
737 \newcommand*{\@glsdefaultplural}{\@glo@text\glspluralsuffix}

\@glsdefaultsort Define command to set default sort.
738 \newcommand*{\@glsdefaultsort}{\@glo@name}

\gls@level Register to increment entry levels.
739 \newcount\gls@level

\newglossaryentry Define \newglossaryentry {<label>} {<key-val list>}. There are two required fields
in <key-val list>: name (or parent) and description. (See above.)
740 \DeclareRobustCommand{\newglossaryentry}[2]{%
  Check to see if this glossary entry has already been defined:
741 \glsdoifnoexists{#1}{%
  Store label
742 \def\@glo@label{#1}%

```

Set up defaults. If the name or description keys are omitted, an error will be generated.

```
743 \let\@glo@name\@glsnoname
```

```
744 \def\@glo@desc{\PackageError{glossaries}{description key required in
```

```
745 \string\newglossaryentry\space for entry '\@glo@label'}{You haven't specified the entry descrip
```

```
746 \def\@glo@descplural{\@glo@desc}%
```

```
747 \def\@glo@type{\glsdefaulttype}%
```

```
748 \def\@glo@symbol{\relax}%
```

```
749 \def\@glo@symbolplural{\@glo@symbol}%
```

```
750 \def\@glo@text{\@glo@name}%
```

```
751 \let\@glo@plural\@glsdefaultplural
```

Using `\let` instead of `\def` to make later comparison avoid expansion issues.
(Thanks to Ulrich Diez for suggesting this.)

```
752 \let\@glo@first\relax
```

```
753 \let\@glo@firstplural\relax
```

Set the default sort:

```
754 \let\@glo@sort\@glsdefaultsort
```

Set the default counter:

```
755 \def\@glo@counter{\@gls@getcounter{\@glo@type}}%
```

```
756 \def\@glo@see{}%
```

```
757 \def\@glo@parent{}%
```

```
758 \def\@glo@prefix{}%
```

```
759 \def\@glo@useri{}%
```

```
760 \def\@glo@userii{}%
```

```
761 \def\@glo@useriii{}%
```

```
762 \def\@glo@useriv{}%
```

```
763 \def\@glo@userv{}%
```

```
764 \def\@glo@uservi{}%
```

Add start hook in case another package wants to add extra keys.

```
765 \@newglossaryentryprehook
```

Extract key-val information from third parameter:

```
766 \setkeys{glossentry}{#2}%
```

Check to see if this glossary type has been defined, if it has, add this label to the relevant list, otherwise generate an error.

```
767 \ifundefined{glolist@\@glo@type}{\PackageError{glossaries}{%
```

```
768 Glossary type '\@glo@type' has not been defined}{%
```

```
769 You need to define a new glossary type, before making entries
```



```

770 in it}}{%
771 \protected@edef\@glo\list@{\csname glo\list@\@glo@type\endcsname}%
772 \expandafter\xdef\csname glo\list@\@glo@type\endcsname{\@glo\list@{#1}},}%
773 }%

Initialise level to 0.
774 \gls@level=0\relax

Has this entry been assigned a parent?
775 \ifx\@glo@parent\@empty

Doesn't have a parent. Set \glo@<label>@parent to empty.
776 \expandafter\gdef\csname glo@#1@parent\endcsname{}%
777 \else

Has a parent. Check to ensure this entry isn't its own parent.
778 \ifthenelse{\equal{#1}{\@glo@parent}}{%
779 \PackageError{glossaries}{Entry '#1' can't be its own parent}{}%
780 \def\@glo@parent{}%
781 \expandafter\gdef\csname glo@#1@parent\endcsname{}%
782 }{%

Check the parent exists:
783 \ifglentryexists{\@glo@parent}{%

Parent exists. Set \glo@<label>@parent.
784 \expandafter\xdef\csname glo@#1@parent\endcsname{\@glo@parent}%

Determine level.
785 \gls@level=\csname glo@\@glo@parent @level\endcsname\relax
786 \advance\gls@level by 1\relax

If name hasn't been specified, use same as the parent name
787 \ifx\@glo@name\@glsnoname
788 \expandafter\let\expandafter\@glo@name
789 \csname glo@\@glo@parent @name\endcsname

If name and plural haven't been specified, use same as the parent
790 \ifx\@glo@plural\@glsdefaultplural
791 \expandafter\let\expandafter\@glo@plural
792 \csname glo@\@glo@parent @plural\endcsname
793 \fi
794 \fi
795 }{%

Parent doesn't exist, so issue an error message and change this entry to have no
parent
796 \PackageError{glossaries}{Invalid parent '@glo@parent'
797 for entry '#1' - parent doesn't exist}{Parent entries
798 must be defined before their children}%
799 \def\@glo@parent{}%
800 \expandafter\gdef\csname glo@#1@parent\endcsname{}%
801 }%
802 }%
803 \fi

```

Set the level for this entry

```
804 \expandafter\xdef\csname glo@#1@level\endcsname{\number\gls@level}%
```

Check if `first` and `firstplural` have been use. If `firstplural` hasn't been specified, but `first` has been specified, then form `firstplural` by appending `\glspluralsuffix` to value of `first` key, otherwise obtain the value from the `plural` key. This now uses `\ifx` instead of `\if` to avoid expansion issues. (Thanks to Ulrich Diez for suggesting this.)

```
805 \ifx\relax\@glo@firstplural
806   \ifx\relax\@glo@first
807     \def\@glo@firstplural{\@glo@plural}%
808     \def\@glo@first{\@glo@text}%
809   \else
810     \def\@glo@firstplural{\@glo@first\glspluralsuffix}%
811   \fi
812 \else
813   \ifx\relax\@glo@first
814     \def\@glo@first{\@glo@text}%
815   \fi
816 \fi
```

Define commands associated with this entry:

```
817 \expandafter
818   \protected\xdef\csname glo@#1@text\endcsname{\@glo@text}%
819 \expandafter
820   \protected\xdef\csname glo@#1@plural\endcsname{\@glo@plural}%
821 \expandafter
822   \protected\xdef\csname glo@#1@first\endcsname{\@glo@first}%
823 \expandafter
824   \protected\xdef\csname glo@#1@firstpl\endcsname{\@glo@firstplural}%
825 \expandafter
826   \protected\xdef\csname glo@#1@type\endcsname{\@glo@type}%
827 \expandafter
828   \protected\xdef\csname glo@#1@counter\endcsname{\@glo@counter}%
829 \expandafter
830   \protected\xdef\csname glo@#1@useri\endcsname{\@glo@useri}%
831 \expandafter
832   \protected\xdef\csname glo@#1@userii\endcsname{\@glo@userii}%
833 \expandafter
834   \protected\xdef\csname glo@#1@useriii\endcsname{\@glo@useriii}%
835 \expandafter
836   \protected\xdef\csname glo@#1@useriv\endcsname{\@glo@useriv}%
837 \expandafter
838   \protected\xdef\csname glo@#1@userv\endcsname{\@glo@userv}%
839 \expandafter
840   \protected\xdef\csname glo@#1@uservi\endcsname{\@glo@uservi}%
841 \@gls@sanitizename
842 \expandafter\protected\xdef\csname glo@#1@name\endcsname{\@glo@name}%
```

The smaller and smallcaps options set the description to `\@glo@first`. Need to

check for this, otherwise it won't get expanded if the description gets sanitized.

```

843 \def\@glo@desc{\@glo@first}%
844 \ifx\@glo@desc\@glo@desc
845   \let\@glo@desc\@glo@first
846 \fi
847 \@gls@sanitizedesc
848 \expandafter\protected@xdef\csname glo@#1@desc\endcsname{\@glo@desc}%
849 \expandafter\protected@xdef\csname glo@#1@descplural\endcsname{\@glo@descplural}%
  Sanitize sort value:
850 \ifx\@glo@sort\@glsdefaultsort
851   \let\@glo@sort\@glo@name
852 \fi
853 \@onelevel@sanitize\@glo@sort
  Set the sort key for this entry:
854 \expandafter\protected@xdef\csname glo@#1@sort\endcsname{\@glo@sort}%
855 \def\@glo@symbol{\@glo@text}%
856 \ifx\@glo@symbol\@glo@symbol
857   \let\@glo@symbol\@glo@text
858 \fi
859 \@gls@sanitizesymbol
860 \expandafter\protected@xdef\csname glo@#1@symbol\endcsname{\@glo@symbol}%
861 \expandafter\protected@xdef\csname glo@#1@symbolplural\endcsname{\@glo@symbolplural}%
  Define an associated boolean variable to determine whether this entry has been
  used yet (needs to be defined globally):
862 \expandafter\gdef\csname glo@#1@flagfalse\endcsname{%
863 \expandafter\global\expandafter
864 \let\csname ifglo@#1@flag\endcsname\iffalse}%
865 \expandafter\gdef\csname glo@#1@flagtrue\endcsname{%
866 \expandafter\global\expandafter
867 \let\csname ifglo@#1@flag\endcsname\iftrue}%
868 \csname glo@#1@flagfalse\endcsname
  Sort out any cross-referencing if required.
869 \ifx\@glo@see\@empty
870 \else
871   \protected@edef\@do@glssee{%
872     \noexpand\@gls@fixbraces\noexpand\@glo@list\@glo@see
873     \noexpand\@nil
874     \noexpand\expandafter\noexpand\@glssee\noexpand\@glo@list{#1}}%
875   \@do@glssee
876 \fi
877 }%
  Determine and store main part of the entry's index format.
878 \@glo@storeentry{#1}%
  Add end hook in case another package wants to add extra keys.
879 \@newglossaryentryposthook
880 }
```

`\@newglossaryentryprehook` Allow extra information to be added to glossary entries:

```
881 \newcommand*{\@newglossaryentryprehook}{}
```

`\@newglossaryentryposthook` Allow extra information to be added to glossary entries:

```
882 \newcommand*{\@newglossaryentryposthook}{}
```

`\@glossaryentryfield` Indicate what command should be used to display each entry in the glossary. (This enables the `glossaries-accsupp` package to use `\accsuppglossaryentryfield` instead.)

```
883 \ifglxindy
884   \newcommand*{\@glossaryentryfield}{\string\@glossaryentryfield}
885 \else
886   \newcommand*{\@glossaryentryfield}{\string\glossaryentryfield}
887 \fi
```

`\@glossarysubentryfield` Indicate what command should be used to display each subentry in the glossary. (This enables the `glossaries-accsupp` package to use `\accsuppglossarysubentryfield` instead.)

```
888 \ifglxindy
889   \newcommand*{\@glossarysubentryfield}{%
890     \string\@glossarysubentryfield}
891 \else
892   \newcommand*{\@glossarysubentryfield}{%
893     \string\glossarysubentryfield}
894 \fi
```

`\@glo@storeentry` Determine the format to write the entry in the glossary output (`.glo`) file. The argument is the entry's label. The result is stored in `\glo@<label>@entry`, where `<label>` is the entry's label. (This doesn't include any formatting or location information.)

```
895 \newcommand{\@glo@storeentry}[1]{%
  Get the sort string and escape any special characters
896 \protected@edef\@glo@sort{\csname glo@#1@sort\endcsname}%
897 \@gls@checkmkidxchars\@glo@sort
  Same again for the name string.
898 \protected@edef\@glo@name{\csname glo@#1@name\endcsname}%
899 \@gls@checkmkidxchars\@glo@name
  Add the font command. (The backslash needs to be escaped for xindy.)
900 \ifglxindy
901   \protected@edef\@glo@name{\string\@gls@namefont{\@glo@name}}%
902 \else
903   \protected@edef\@glo@name{\string\gls@namefont{\@glo@name}}%
904 \fi
  Get the description string and escape any special characters
905 \protected@edef\@glo@desc{\csname glo@#1@desc\endcsname}%
906 \@gls@checkmkidxchars\@glo@desc
```

Same again for the symbol

```
907 \protected@edef\@glo@symbol{\csname glo@#1@symbol\endcsname}%  
908 \@gls@checkmkidxchars\@glo@symbol
```

Escape any special characters in the prefix

```
909 \@gls@checkmkidxchars\@glo@prefix
```

Get the parent, if one exists

```
910 \edef\@glo@parent{\csname glo@#1@parent\endcsname}%
```

Write the information to the glossary file.

```
911 \ifglxsindy
```

Store using xindy syntax.

```
912 \ifx\@glo@parent\@empty
```

Entry doesn't have a parent

```
913 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%  
914 (\string"\@glo@sort\string" %  
915 \string"\@glo@prefix\@glossaryentryfield{#1}\@glo@name  
916 }\@glo@desc}\@glo@symbol}\string") %  
917 }%  
918 \else
```

Entry has a parent

```
919 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%  
920 \csname glo@\@glo@parent @index\endcsname  
921 (\string"\@glo@sort\string" %  
922 \string"\@glo@prefix\@glossarysubentryfield%  
923 { \csname glo@#1@level\endcsname}{#1}\@glo@name  
924 }\@glo@desc}\@glo@symbol}\string") %  
925 }%  
926 \fi  
927 \else
```

Store using makeindex syntax.

```
928 \ifx\@glo@parent\@empty
```

Sanitize \@glo@prefix

```
929 \@onelevel@sanitize\@glo@prefix
```

Entry doesn't have a parent

```
930 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%  
931 \@glo@sort\@gls@actualchar\@glo@prefix  
932 \@glossaryentryfield{#1}\@glo@name}\@glo@desc  
933 }\@glo@symbol}%  
934 }%  
935 \else
```

Entry has a parent

```
936 \expandafter\protected@xdef\csname glo@#1@index\endcsname{%  
937 \csname glo@\@glo@parent @index\endcsname\@gls@levelchar  
938 \@glo@sort\@gls@actualchar\@glo@prefix
```

```

939     \@glossarysubentryfield
940     {\csname glo@#1@level\endcsname}{#1}{\@glo@name}{\@glo@desc
941     }{\@glo@symbol}}%
942   }%
943   \fi
944 \fi
945 }

```

4.8 Resetting and unsetting entry flags

Each glossary entry is assigned a conditional of the form `\ifglo@<label>@flag` which determines whether or not the entry has been used (see also `\ifglsused` defined below). These flags can be set and unset using the following macros:

The command `\glsreset{<label>}` can be used to set the entry flag to indicate that it hasn't been used yet. The required argument is the entry label.

`\glsreset`

```

946 \newcommand*{\glsreset}[1]{%
947 \glsdoifexists{#1}{%
948 \expandafter\global\csname glo@#1@flagfalse\endcsname}}

```

As above, but with only a local effect:

`\glslocalreset`

```

949 \newcommand*{\glslocalreset}[1]{%
950 \glsdoifexists{#1}{%
951 \expandafter\let\csname ifglo@#1@flag\endcsname\iffalse}}

```

The command `\glsunset{<label>}` can be used to set the entry flag to indicate that it has been used. The required argument is the entry label.

`\glsunset`

```

952 \newcommand*{\glsunset}[1]{%
953 \glsdoifexists{#1}{%
954 \expandafter\global\csname glo@#1@flagtrue\endcsname}}

```

As above, but with only a local effect:

`\glslocalunset`

```

955 \newcommand*{\glslocalunset}[1]{%
956 \glsdoifexists{#1}{%
957 \expandafter\let\csname ifglo@#1@flag\endcsname\iftrue}}

```

Reset all entries for the named glossaries (supplied in a comma-separated list).

Syntax: `\glsresetall[<glossary-list>]`

`\glsresetall`

```

958 \newcommand*{\glsresetall}[1][\@glo@types]{%
959 \forallglsentries[#1]{\@glsentry}{%
960 \glsreset{\@glsentry}}}

```

As above, but with only a local effect:

`\glslocalresetall`

```
961 \newcommand*{\glslocalresetall}[1][\@glo@types]{%
962 \forallglsentries[#1]{\@glsentry}{%
963 \glslocalreset{\@glsentry}}}
```

Unset all entries for the named glossaries (supplied in a comma-separated list).

Syntax: `\glsunsetall` [*<glossary-list>*]

`\glsunsetall`

```
964 \newcommand*{\glsunsetall}[1][\@glo@types]{%
965 \forallglsentries[#1]{\@glsentry}{%
966 \glsunset{\@glsentry}}}
```

As above, but with only a local effect:

`\glslocalunsetall`

```
967 \newcommand*{\glslocalunsetall}[1][\@glo@types]{%
968 \forallglsentries[#1]{\@glsentry}{%
969 \glslocalunset{\@glsentry}}}
```

4.9 Loading files containing glossary entries

Glossary entries can be defined in an external file. These external files can contain `\newglossaryentry` and `\newacronym` commands.²⁰

`\loadglsentries` [*<type>*] [*<filename>*]

This command will input the file using `\input`. The optional argument specifies to which glossary the entries should be assigned if they haven't used the `type` key. If the optional argument is not specified, the default glossary is used. Only those entries used in the document (via `\glslink`, `\gls`, `\glspl` and uppercase variants or `\glsadd` and `\glsaddall` will appear in the glossary). The mandatory argument is the filename (with or without `.tex` extension).

`\loadglsentries`

```
970 \newcommand*{\loadglsentries}[2][\@gls@default]{%
971 \let\@gls@default\glsdefaulttype
972 \def\glsdefaulttype{#1}\input{#2}%
973 \let\glsdefaulttype\@gls@default}
```

`\loadglsentries` can only be used in the preamble:

```
974 \@onlypreamble{\loadglsentries}
```

²⁰and any other valid L^AT_EX code that can be used in the preamble.

4.10 Using glossary entries in the text

Any term that has been defined using `\newglossaryentry` (or `\newacronym`) can be displayed in the text (i.e. outside of the glossary) using one of the commands defined in this section. Unless you use `\glslink`, the way the term appears in the text is determined by `\glsdisplayfirst` (if it is the first time the term has been used) or `\glsdisplay` (for subsequent use). Any formatting commands (such as `\textbf`) is governed by `\glstextformat`. By default this just displays the link text “as is”.

`\glstextformat`

```
975 \newcommand*{\glstextformat}[1]{#1}
```

The first time an entry is used, the way in which it is displayed is governed by `\glsdisplayfirst`. This takes four parameters: `#1` will be the value of the entry’s `first` or `firstplural` key, `#2` will be the value of the entry’s `description` key, `#3` will be the value of the entry’s `symbol` key and `#4` is additional text supplied by the final optional argument to commands like `\gls` and `\glsp1`. The default is to display the first parameter followed by the additional text.

`\glsdisplayfirst`

```
976 \newcommand*{\glsdisplayfirst}[4]{#1#4}
```

After the first use, the entry is displayed according to the format of `\glsdisplay`. Again, it takes four parameters: `#1` will be the value of the entry’s `text` or `plural` key, `#2` will be the value of the entry’s `description` key, `#3` will be the value of the entry’s `symbol` key and `#4` is additional text supplied by the final optional argument to commands like `\gls` and `\glsp1`.

`\glsdisplay`

```
977 \newcommand*{\glsdisplay}[4]{#1#4}
```

When a new glossary is created it uses `\glsdisplayfirst` and `\glsdisplay` as the default way of displaying its entry in the text. This can be changed for the entries belonging to an individual glossary using `\defglsdisplay` and `\defglsdisplayfirst`.

```
\defglsdisplay[⟨type⟩]{⟨definition⟩}
```

The glossary type is given by `⟨type⟩` (the default glossary if omitted) and `⟨definition⟩` should have at most `#1`, `#2`, `#3` and `#4`. These represent the same arguments as those described for `\glsdisplay`.

`\defglsdisplay`

```
978 \newcommand*{\defglsdisplay}[2][\glsdefaultttype]{%
979 \expandafter\def\csname gls@#1@display\endcsname##1##2##3##4{#2}}
```

```
\defglsdisplayfirst[⟨type⟩]{⟨definition⟩}
```


The glossary type is given by $\langle type \rangle$ (the default glossary if omitted) and $\langle definition \rangle$ should have at most #1, #2, #3 and #4. These represent the same arguments as those described for `\glsdisplayfirst`.

`\defglsdisplayfirst`

```
980 \newcommand*{\defglsdisplayfirst}[2][\glsdefaultttype]{%
981 \expandafter\def\csname gls@#1@displayfirst\endcsname##1##2##3##4{#2}}
```

4.10.1 Links to glossary entries

The links to glossary entries all have a first optional argument that can be used to change the format and counter of the associated entry number. Except for `\glslink`, the commands like `\gls` have a final optional argument that can be used to insert additional text in the link (this will usually be appended, but can be redefined using `\defglsdisplay` and `\defglsdisplayfirst`). It goes against the L^AT_EX norm to have an optional argument after the mandatory arguments, but it makes more sense to write, say, `\gls{label}[s]` rather than, say, `\gls[append=s]{label}`. Since these control sequences are defined to include the final square bracket, spaces will be ignored after them. This is likely to lead to confusion as most users would not expect, say, `\gls{label}` to ignore following spaces, so `\new@ifnextchar` from the package is required.

The following keys can be used in the first optional argument. The counter key checks that the value is the name of a valid counter.

```
982 \define@key{glslink}{counter}{%
983 \@ifundefined{c@#1}{\PackageError{glossaries}{There is no counter
984 called '#1'}{The counter key should have the name of a valid
985 counter as its value}}{%
986 \def\@gls@counter{#1}}}
```

The value of the format key should be the name of a command (without the initial backslash) that has a single mandatory argument which can be used to format the associated entry number.

```
987 \define@key{glslink}{format}{%
988 \def\@glsnumberformat{#1}}
```

The hyper key is a boolean key, it can either have the value true or false, and indicates whether or not to make a hyperlink to the relevant glossary entry. If hyper is false, an entry will still be made in the glossary, but the given text won't be a hyperlink.

```
989 \define@boolkey{glslink}{hyper}[true]{}
```

Syntax:

```
\glslink[ $\langle options \rangle$ ]{ $\langle label \rangle$ }{ $\langle text \rangle$ }
```

Display $\langle text \rangle$ in the document, and add the entry information for $\langle label \rangle$ into the relevant glossary. The optional argument should be a key value list using the `glslink` keys defined above.

There is also a starred version:

`\glslink*[\langle options \rangle]{\langle label \rangle}{\langle text \rangle}`

which is equivalent to `\glslink[hyper=false,\langle options \rangle]{\langle label \rangle}{\langle text \rangle}`

First determine whether or not we are using the starred version:

```
\glslink
990 \newcommand{\glslink}{%
991 \ifstar\sgls@link\gls@link}

\sgls@link The starred version of \glslink calls the unstarred version with hyperlinks disabled.
992 \newcommand*\sgls@link[1][]{\gls@link[hyper=false,#1]}

\gls@link The unstarred version of \glslink checks for the existence of the term. The main part of the business is in \gls@link which shouldn't check if the term is defined as it's called by \gls etc which also perform that check.
993 \newcommand*\gls@link[3][]{%
994 \ifglentryexists{#2}%
995 {%
996 \gls@link[#1]{#2}{#3}%
997 }{%
998 \PackageError{glossaries}{Glossary entry '#2' has not been
999 defined}{You need to define a glossary entry before you
1000 can use it.}%
    Display the specified text. (The entry doesn't exist so there's nothing to link it to.)
1001 \glstextformat{#3}%
1002 }%
1003 }

\gls@link
1004 \def\gls@link[#1]#2#3{%
    Inserting \leavevmode suggested by Donald Arseneau (avoids problem with tabularx).
1005 \leavevmode
1006 \def\glslabel{#2}%
1007 \def\glsnumberformat{glsnumberformat}%
1008 \edef\gls@counter{\csname glo@#2@counter\endcsname}%
1009 \KV@glslink@hypertrue
1010 \setkeys{glslink}{#1}%
1011 \edef\theglentrycounter{\expandafter\noexpand
1012 \csname the\gls@counter\endcsname}%

1013 \@do@wrglossary{#2}%
1014 \ifKV@glslink@hyper
1015 \glslink{glo:#2}{\glstextformat{#3}}%
1016 \else
1017 \glstextformat{#3}\relax
```

```

1018     \fi
1019 }

```

Set the formatting information in the format required by `makeindex`. The first argument is the format specified by the user (via the format key), the second argument is the name of the counter used to indicate the location and the third argument is a control sequence which stores the required format.

`\@set@glo@numformat`

```

1020 \def\@set@glo@numformat#1#2#3{%
1021 \expandafter\@glo@check@mkidxrangechar#3\@nil
1022 \protected@edef#1{\@glo@prefix setentrycounter{#2}%
1023 \expandafter\string\csname\@glo@suffix\endcsname}%
1024 \@gls@checkmkidxchars#1}

```

Check to see if the given string starts with a (or). If it does set `\@glo@prefix` to the starting character, and `\@glo@suffix` to the rest (or `glsnumberformat` if there is nothing else), otherwise set `\@glo@prefix` to nothing and `\@glo@suffix` to all of it.

```

1025 \def\@glo@check@mkidxrangechar#1#2\@nil{%
1026 \if#1(\relax
1027   \def\@glo@prefix{(%
1028   \if\relax#2\relax
1029     \def\@glo@suffix{glsnumberformat}%
1030   \else
1031     \def\@glo@suffix{#2}%
1032   \fi
1033 \else
1034   \if#1)\relax
1035     \def\@glo@prefix{)%
1036     \if\relax#2\relax
1037       \def\@glo@suffix{glsnumberformat}%
1038     \else
1039       \def\@glo@suffix{#2}%
1040     \fi
1041   \else
1042     \def\@glo@prefix{}\def\@glo@suffix{#1#2}%
1043   \fi
1044 \fi}

```

`\@gls@escbsdq` Escape backslashes and double quote marks. The argument must be a control sequence.

```

1045 \newcommand*{\@gls@escbsdq}[1]{%
1046   \def\@gls@checkedmkidx{}%
1047   \let\gls@xdystring=#1\relax
1048   \@onelevel@sanitize\gls@xdystring
1049   \edef\do@gls@xdycheckbackslash{%
1050     \noexpand\@gls@xdycheckbackslash\gls@xdystring\noexpand\@nil
1051     \@backslashchar\@backslashchar\noexpand\null}%

```

```

1052 \do@gl@xdycheckbackslash
1053 \expandafter\@gl@updatechecked\@gl@checkedmkidx{\gl@xdystring}%
1054 \def\@gl@checkedmkidx{}%
1055 \expandafter\@gl@xdycheckquote\gl@xdystring\@nil""\null
1056 \expandafter\@gl@updatechecked\@gl@checkedmkidx{\gl@xdystring}%
1057 \let#1=\gl@xdystring
1058 }

```

Catch special characters(argument must be a control sequence):

\@gl@checkmkidxchars

```

1059 \newcommand{\@gl@checkmkidxchars}[1]{%
1060 \ifgl@xindy
1061 \@gl@escbsdq{#1}%
1062 \else
1063 \def\@gl@checkedmkidx{}%
1064 \expandafter\@gl@checkquote#1\@nil""\null
1065 \expandafter\@gl@updatechecked\@gl@checkedmkidx{#1}%
1066 \def\@gl@checkedmkidx{}%
1067 \expandafter\@gl@checkescquote#1\@nil""\null
1068 \expandafter\@gl@updatechecked\@gl@checkedmkidx{#1}%
1069 \def\@gl@checkedmkidx{}%
1070 \expandafter\@gl@checkescactual#1\@nil\?\?\null
1071 \expandafter\@gl@updatechecked\@gl@checkedmkidx{#1}%
1072 \def\@gl@checkedmkidx{}%
1073 \expandafter\@gl@checkactual#1\@nil??\null
1074 \expandafter\@gl@updatechecked\@gl@checkedmkidx{#1}%
1075 \def\@gl@checkedmkidx{}%
1076 \expandafter\@gl@checkbar#1\@nil||\null
1077 \expandafter\@gl@updatechecked\@gl@checkedmkidx{#1}%
1078 \def\@gl@checkedmkidx{}%
1079 \expandafter\@gl@checkescbar#1\@nil\\|\null
1080 \expandafter\@gl@updatechecked\@gl@checkedmkidx{#1}%
1081 \def\@gl@checkedmkidx{}%
1082 \expandafter\@gl@checklevel#1\@nil!!\null
1083 \expandafter\@gl@updatechecked\@gl@checkedmkidx{#1}%
1084 \fi
1085 }

```

Update the control sequence and strip trailing \@nil:

\@gl@updatechecked

```

1086 \def\@gl@updatechecked#1\@nil#2{\def#2{#1}}

```

\@gl@tmpb Define temporary token

```

1087 \newtoks\@gl@tmpb

```

\@gl@checkquote Replace " with "" since " is a makeindex special character.

```

1088 \def\@gl@checkquote#1"#2"#3\null{%
1089 \@gl@tmpb=\expandafter{\@gl@checkedmkidx}%

```

```

1090 \toks@={#1}%
1091 \ifx\null#2\null
1092 \ifx\null#3\null
1093 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1094 \def\@gls@checkquote{\relax}%
1095 \else
1096 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1097 \@gls@quotechar\@gls@quotechar\@gls@quotechar\@gls@quotechar}%
1098 \def\@gls@checkquote{\@gls@checkquote#3\null}%
1099 \fi
1100 \else
1101 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1102 \@gls@quotechar\@gls@quotechar}%
1103 \ifx\null#3\null
1104 \def\@gls@checkquote{\@gls@checkquote#2""\null}%
1105 \else
1106 \def\@gls@checkquote{\@gls@checkquote#2"#3\null}%
1107 \fi
1108 \fi
1109 \@gls@checkquote}

```

\@gls@checkescquote Do the same for \":

```

1110 \def\@gls@checkescquote#1"#2"#3\null{%
1111 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1112 \toks@={#1}%
1113 \ifx\null#2\null
1114 \ifx\null#3\null
1115 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1116 \def\@gls@checkescquote{\relax}%
1117 \else
1118 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1119 \@gls@quotechar\string"\@gls@quotechar
1120 \@gls@quotechar\string"\@gls@quotechar}%
1121 \def\@gls@checkescquote{\@gls@checkescquote#3\null}%
1122 \fi
1123 \else
1124 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1125 \@gls@quotechar\string"\@gls@quotechar}%
1126 \ifx\null#3\null
1127 \def\@gls@checkescquote{\@gls@checkescquote#2""\null}%
1128 \else
1129 \def\@gls@checkescquote{\@gls@checkescquote#2"#3\null}%
1130 \fi
1131 \fi
1132 \@gls@checkescquote}

```

\@gls@checkescactual Similarly for \? (which is replaces @ as makeindex's special character):

```

1133 \def\@gls@checkescactual#1?#2?#3\null{%
1134 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1135 \toks@={#1}%

```

```

1136 \ifx\null#2\null
1137 \ifx\null#3\null
1138 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1139 \def\@gls@checkescactual{\relax}%
1140 \else
1141 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1142 \gls@quotechar\string"\@gls@actualchar
1143 \gls@quotechar\string"\@gls@actualchar}%
1144 \def\@gls@checkescactual{\@gls@checkescactual#3\null}%
1145 \fi
1146 \else
1147 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1148 \gls@quotechar\string"\@gls@actualchar}%
1149 \ifx\null#3\null
1150 \def\@gls@checkescactual{\@gls@checkescactual#2\?\?\null}%
1151 \else
1152 \def\@gls@checkescactual{\@gls@checkescactual#2\?#3\null}%
1153 \fi
1154 \fi
1155 \@gls@checkescactual}

```

\@gls@checkescbar Similarly for \|:

```

1156 \def\@gls@checkescbar#1\|#2\|#3\null{%
1157 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1158 \toks@={#1}%
1159 \ifx\null#2\null
1160 \ifx\null#3\null
1161 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1162 \def\@gls@checkescbar{\relax}%
1163 \else
1164 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1165 \gls@quotechar\string"\@gls@encapchar
1166 \gls@quotechar\string"\@gls@encapchar}%
1167 \def\@gls@checkescbar{\@gls@checkescbar#3\null}%
1168 \fi
1169 \else
1170 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1171 \gls@quotechar\string"\@gls@encapchar}%
1172 \ifx\null#3\null
1173 \def\@gls@checkescbar{\@gls@checkescbar#2\|\|\null}%
1174 \else
1175 \def\@gls@checkescbar{\@gls@checkescbar#2\|#3\null}%
1176 \fi
1177 \fi
1178 \@gls@checkescbar}

```

\@gls@checkesclevel Similarly for \!:

```

1179 \def\@gls@checkesclevel#1\!#2\!#3\null{%
1180 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1181 \toks@={#1}%

```

```

1182 \ifx\null#2\null
1183 \ifx\null#3\null
1184 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1185 \def\@gls@checkesclevel{\relax}%
1186 \else
1187 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1188 \@gls@quotechar\string"\@gls@levelchar
1189 \@gls@quotechar\string"\@gls@levelchar}%
1190 \def\@gls@checkesclevel{\@gls@checkesclevel#3\null}%
1191 \fi
1192 \else
1193 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1194 \@gls@quotechar\string"\@gls@levelchar}%
1195 \ifx\null#3\null
1196 \def\@gls@checkesclevel{\@gls@checkesclevel#2\!\!\null}%
1197 \else
1198 \def\@gls@checkesclevel{\@gls@checkesclevel#2\!#3\null}%
1199 \fi
1200 \fi
1201 \@gls@checkesclevel}

\@gls@checkbar and for |:
1202 \def\@gls@checkbar#1|#2|#3\null{%
1203 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1204 \toks@={#1}%
1205 \ifx\null#2\null
1206 \ifx\null#3\null
1207 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1208 \def\@gls@checkbar{\relax}%
1209 \else
1210 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1211 \@gls@quotechar\@gls@encapchar\@gls@quotechar\@gls@encapchar}%
1212 \def\@gls@checkbar{\@gls@checkbar#3\null}%
1213 \fi
1214 \else
1215 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1216 \@gls@quotechar\@gls@encapchar}%
1217 \ifx\null#3\null
1218 \def\@gls@checkbar{\@gls@checkbar#2|\!\null}%
1219 \else
1220 \def\@gls@checkbar{\@gls@checkbar#2|#3\null}%
1221 \fi
1222 \fi
1223 \@gls@checkbar}

\@gls@checklevel and for !:
1224 \def\@gls@checklevel#1!#2!#3\null{%
1225 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1226 \toks@={#1}%
1227 \ifx\null#2\null

```

```

1228 \ifx\null#3\null
1229 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1230 \def\@@gls@checklevel{\relax}%
1231 \else
1232 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1233 \@gls@quotechar\@gls@levelchar\@gls@quotechar\@gls@levelchar}%
1234 \def\@@gls@checklevel{\@gls@checklevel#3\null}%
1235 \fi
1236 \else
1237 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1238 \@gls@quotechar\@gls@levelchar}%
1239 \ifx\null#3\null
1240 \def\@@gls@checklevel{\@gls@checklevel#2!\null}%
1241 \else
1242 \def\@@gls@checklevel{\@gls@checklevel#2!#3\null}%
1243 \fi
1244 \fi
1245 \@@gls@checklevel}

```

\@gls@checkactual and for ?:

```

1246 \def\@gls@checkactual#1?#2?#3\null{%
1247 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1248 \toks@={#1}%
1249 \ifx\null#2\null
1250 \ifx\null#3\null
1251 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%
1252 \def\@@gls@checkactual{\relax}%
1253 \else
1254 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1255 \@gls@quotechar\@gls@actualchar\@gls@quotechar\@gls@actualchar}%
1256 \def\@@gls@checkactual{\@gls@checkactual#3\null}%
1257 \fi
1258 \else
1259 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1260 \@gls@quotechar\@gls@actualchar}%
1261 \ifx\null#3\null
1262 \def\@@gls@checkactual{\@gls@checkactual#2??\null}%
1263 \else
1264 \def\@@gls@checkactual{\@gls@checkactual#2?#3\null}%
1265 \fi
1266 \fi
1267 \@@gls@checkactual}

```

\@gls@xdycheckquote As before but for use with xindy

```

1268 \def\@gls@xdycheckquote#1"#2"#3\null{%
1269 \@gls@tmpb=\expandafter{\@gls@checkedmkidx}%
1270 \toks@={#1}%
1271 \ifx\null#2\null
1272 \ifx\null#3\null
1273 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@}%

```



```

1274 \def\@gls@xdycheckquote{\relax}%
1275 \else
1276 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1277 \string"\string"}%
1278 \def\@gls@xdycheckquote{\@gls@xdycheckquote#3\null}%
1279 \fi
1280 \else
1281 \edef\@gls@checkedmkidx{\the\@gls@tmpb\the\toks@
1282 \string"}%
1283 \ifx\null#3\null
1284 \def\@gls@xdycheckquote{\@gls@xdycheckquote#2""\null}%
1285 \else
1286 \def\@gls@xdycheckquote{\@gls@xdycheckquote#2"#3\null}%
1287 \fi
1288 \fi
1289 \@gls@xdycheckquote
1290 }

```

\@gls@xdycheckbackslash Need to escape all backslashes for xindy. Define command that will define \@gls@xdycheckbackslash

```

1291 \edef\def\@gls@xdycheckbackslash{%
1292 \noexpand\def\noexpand\@gls@xdycheckbackslash##1\@backslashchar
1293 ##2\@backslashchar##3\noexpand\null{%
1294 \noexpand\@gls@tmpb=\noexpand\expandafter
1295 {\noexpand\@gls@checkedmkidx}%
1296 \noexpand\toks@={##1}%
1297 \noexpand\ifx\noexpand\null##2\noexpand\null
1298 \noexpand\ifx\noexpand\null##3\noexpand\null
1299 \noexpand\edef\noexpand\@gls@checkedmkidx{%
1300 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@}%
1301 \noexpand\def\noexpand\@gls@xdycheckbackslash{\relax}%
1302 \noexpand\else
1303 \noexpand\edef\noexpand\@gls@checkedmkidx{%
1304 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
1305 \@backslashchar\@backslashchar\@backslashchar\@backslashchar}%
1306 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
1307 \noexpand\@gls@xdycheckbackslash##3\noexpand\null}%
1308 \noexpand\fi
1309 \noexpand\else
1310 \noexpand\edef\noexpand\@gls@checkedmkidx{%
1311 \noexpand\the\noexpand\@gls@tmpb\noexpand\the\noexpand\toks@
1312 \@backslashchar\@backslashchar}%
1313 \noexpand\ifx\noexpand\null##3\noexpand\null
1314 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
1315 \noexpand\@gls@xdycheckbackslash##2\@backslashchar
1316 \@backslashchar\noexpand\null}%
1317 \noexpand\else
1318 \noexpand\def\noexpand\@gls@xdycheckbackslash{%
1319 \noexpand\@gls@xdycheckbackslash##2\@backslashchar
1320 ##3\noexpand\null}%

```

```

1321 \noexpand\fi
1322 \noexpand\fi
1323 \noexpand\@gls@xdycheckbackslash
1324 }%
1325 }

```

Now go ahead and define \gls@xdycheckbackslash

```

1326 \def@gls@xdycheckbackslash

```

\@glslink If \hyperlink is not defined \@glslink ignores its first argument and just does the second argument, otherwise it is equivalent to \hyperlink.

```

1327 \@ifundefined{hyperlink}{%
1328 \gdef@glslink#1#2{#2}%
1329 }{%
1330 \gdef@glslink#1#2{\hyperlink{#1}{#2}}%
1331 }

```

\@glstarget If \hypertarget is not defined, \@glstarget ignores its first argument and just does the second argument, otherwise it is equivalent to \hypertarget.

```

1332 \newlength@gls@tmplen
1333 \@ifundefined{hypertarget}{%
1334 \gdef@glstarget#1#2{#2}%
1335 }{%
1336 \gdef@glstarget#1#2{%
1337 \settoheight@gls@tmplen{#2}%
1338 \raisebox@gls@tmplen{\hypertarget{#1}{}}{#2}%
1339 }

```

Glossary hyperlinks can be disabled using \glsdisablehyper (effect can be localised):

\glsdisablehyper

```

1340 \newcommand@glsdisablehyper{%
1341 \renewcommand*@glslink[2]{##2}%
1342 \renewcommand*@glstarget[2]{##2}}

```

Glossary hyperlinks can be enabled using \glsenablehyper (effect can be localised):

\glsenablehyper

```

1343 \newcommand@glsenablehyper{%
1344 \renewcommand*@glslink[2]{\hyperlink{##1}{##2}}%
1345 \renewcommand*@glstarget[2]{%
1346 \settoheight@gls@tmplen{##2}%
1347 \raisebox@gls@tmplen{\hypertarget{##1}{}}{##2}}

```

Syntax:

\gls[*<options>*]{*<label>*}[*<insert text>*]

Link to glossary entry using singular form. The link text is taken from the value of the `text` or `first` keys used when the entry was defined.

The first optional argument is a key-value list, the same as `\glslink`, the mandatory argument is the entry label. After the mandatory argument, there is another optional argument to insert extra text in the link text (the location of the inserted text is governed by `\glsdisplay` and `\glsdisplayfirst`). As with `\glslink` there is a starred version which is the same as the unstarred version but with the `hyper` key set to `false`. (Additional options can also be specified in the first optional argument.)

First determine if we are using the starred form:

```
\gls
1348 \newcommand*{\gls}{\@ifstar\@sgls\@gls}

Define the starred form:

\sgls
1349 \newcommand*{\@sgls}[1] [] {\@gls[hyper=false,#1]}

Defined the un-starred form. Need to determine if there is a final optional argument

\@gls
1350 \newcommand*{\@gls}[2] [] {%
1351 \new@ifnextchar[{\@gls@{#1}{#2}}{\@gls@{#1}{#2} []}}

\@gls@ Read in the final optional argument:
1352 \def\@gls@#1#2[#3] {%
1353 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
    Save options in \@gls@link@opts and label in \@gls@link@label
1354 \def\@gls@link@opts{#1}%
1355 \def\@gls@link@label{#2}%

    Determine what the link text should be (this is stored in \@glo@text)
1356 \ifglsused{#2}%
1357 {%
1358   \def\@glo@text{%
1359     \csname gls@\@glo@type @display\endcsname
1360     {\glsentrytext{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}%
1361   }%
1362 {%
1363   \def\@glo@text{%
1364     \csname gls@\@glo@type @displayfirst\endcsname
1365     {\glsentryfirst{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}%
1366   }%

    Call \@gls@link. If footnote package option has been used and the glossary type
    is \acronymtype, suppress hyperlink for first use. Likewise if the hyperfirst=false
    package option is used.
```

```

1367 \ifglsused{#2}{%
1368   \@gls@link[#1]{#2}{\@glo@text}%
1369 }{%
1370   \gls@checkisacronymlist\@glo@type
1371   \ifthenelse{(\boolean{@glsisacronymlist})\AND
1372     \boolean{glsacrfootnote}) \OR \NOT\boolean{glshyperfirst}}{%
1373     \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
1374   }{%
1375     \@gls@link[#1]{#2}{\@glo@text}%
1376   }%
1377 }%

  Indicate that this entry has now been used
1378 \glsunset{#2}}%
1379 }

```

`\Gls` behaves like `\gls`, but the first letter of the link text is converted to uppercase (note that if the first letter has an accent, the accented letter will need to be grouped when you define the entry). It is mainly intended for terms that start a sentence:

```

\Gls
1380 \newcommand*{\Gls}{\@ifstar\@sGls\@Gls}

  Define the starred form:
1381 \newcommand*{\@sGls}[1][]{\@Gls[hyper=false,#1]}

  Defined the un-starred form. Need to determine if there is a final optional argu-
  ment
1382 \newcommand*{\@Gls}[2][]{%
1383   \new@ifnextchar[{\@Gls@{#1}{#2}}{\@Gls@{#1}{#2}}[]]}

\@Gls@  Read in the final optional argument:
1384 \def\@Gls@#1#2[#3]{%
1385   \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%

  Save options in \@gls@link@opts and label in \@gls@link@label
1386 \def\@gls@link@opts{#1}%
1387 \def\@gls@link@label{#2}%
1388 \def\glslabel{#2}%

  Determine what the link text should be (this is stored in \@glo@text)
1389 \ifglsused{#2}%
1390 {%
1391   \protected@edef\@glo@text{%
1392     \csname gls@\@glo@type @display\endcsname
1393     {\glsentrytext{#2}}{\glsentrydesc{#2}}}%
1394     {\glsentrysymbol{#2}}{#3}}%
1395 }%
1396 {%
1397   \protected@edef\@glo@text{%

```

```

1398 \csname gls@\glo@type @displayfirst\endcsname
1399 {\glsentryfirst{#2}}{\glsentrydesc{#2}}%
1400 {\glsentrysymbol{#2}}{#3}}%
1401 }%

```

Call `\gls@link` If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```

1402 \ifglsused{#2}{%
1403 \gls@link[#1]{#2}{%
1404 \expandafter\makefirstuc\expandafter{\glo@text}}%
1405 }{%
1406 \gls@checkisacronymlist\glo@type
1407 \ifthenelse{(\boolean{@glsisacronymlist})\AND
1408 \boolean{glsacrfootnote}) \OR \NOT\boolean{glshyperfirst}}{%
1409 \gls@link[#1,hyper=false]{#2}{%
1410 \expandafter\makefirstuc\expandafter{\glo@text}}%
1411 }{%
1412 \gls@link[#1]{#2}{%
1413 \expandafter\makefirstuc\expandafter{\glo@text}}%
1414 }%
1415 }%

```

Indicate that this entry has now been used

```

1416 \glsunset{#2}}%
1417 }

```

`\GLS` behaves like `\gls`, but the link text is converted to uppercase:

`\GLS`

```

1418 \newcommand*{\GLS}{\ifstar\sGLS\@GLS}

```

Define the starred form:

```

1419 \newcommand*{\sGLS}[1][]{\@GLS[hyper=false,#1]}

```

Defined the un-starred form. Need to determine if there is a final optional argument

```

1420 \newcommand*{\@GLS}[2][]{%
1421 \new@ifnextchar[{\@GLS@{#1}{#2}}{\@GLS@{#1}{#2}[]}]

```

`\@GLS@` Read in the final optional argument:

```

1422 \def\@GLS@#1#2[#3]{%
1423 \glsdoifexists{#2}{\edef\glo@type{\glsentrytype{#2}}%
    Save options in \gls@link@opts and label in \gls@link@label
1424 \def\gls@link@opts{#1}%
1425 \def\gls@link@label{#2}%

```

Determine what the link text should be (this is stored in `\glo@text`).

```

1426 \ifglsused{#2}{\def\glo@text{%
1427 \csname gls@\glo@type @display\endcsname

```

```

1428 {\glsentrytext{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}}%
1429 \def\@glo@text{%
1430 \csname gls@\@glo@type @displayfirst\endcsname
1431 {\glsentryfirst{#2}}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{#3}}}%

    Call \@gls@link If footnote package option has been used and the glossary type
    is \acronymtype, suppress hyperlink for first use. Likewise if the hyperfirst=false
    package option is used.

1432 \ifglsused{#2}{%
1433   \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}}%
1434 }{%
1435   \gls@checkisacronymlist\@glo@type
1436   \ifthenelse{(\boolean{\@glsisacronymlist})\AND
1437     \boolean{\@glsacrfootnote}) \OR \NOT\boolean{\@gls@hyperfirst}}{%
1438     \@gls@link[#1,hyper=false]{#2}{\MakeUppercase{\@glo@text}}}%
1439   }{%
1440     \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}}%
1441   }%
1442 }%

    Indicate that this entry has now been used

1443 \glsunset{#2}}%
1444 }

```

\glspl behaves in the same way as \gls except it uses the plural form.

\glspl

```

1445 \newcommand*{\glspl}{\@ifstar\@sglspl\@glspl}

    Define the starred form:

1446 \newcommand*{\@sglspl}[1][\@glspl[hyper=false,#1]]{

    Defined the un-starred form. Need to determine if there is a final optional argu-
    ment

1447 \newcommand*{\@glspl}[2][\@glspl@{#1}{#2}]{\@glspl@{#1}{#2}}{}}
1448 \new@ifnextchar[\@glspl@{#1}{#2}]{\@glspl@{#1}{#2}}{}}

```

\@glspl@ Read in the final optional argument:

```

1449 \def\@glspl@#1#2[#3]{%
1450 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%

    Save options in \@gls@link@opts and label in \@gls@link@label

1451 \def\@gls@link@opts{#1}%
1452 \def\@gls@link@label{#2}%

    Determine what the link text should be (this is stored in \@glo@text)

1453 \ifglsused{#2}%
1454 {%
1455   \def\@glo@text{%
1456     \csname gls@\@glo@type @display\endcsname
1457     {\glsentryplural{#2}}{\glsentrydescplural{#2}}}%

```

```

1458      {\glsentrysymbolplural{#2}{#3}}%
1459 }%
1460 {%
1461   \def\@glo@text{%
1462     \csname gls@\@glo@type @displayfirst\endcsname
1463     {\glsentryfirstplural{#2}}{\glsentrydescplural{#2}}%
1464     {\glsentrysymbolplural{#2}{#3}}%
1465 }%

```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```

1466 \ifglsused{#2}{%
1467   \@gls@link[#1]{#2}{\@glo@text}%
1468 }{%
1469   \gls@checkisacronymlist\@glo@type
1470   \ifthenelse{(\boolean{@glsisacronymlist})\AND
1471     \boolean{glsacrfootnote}) \OR \NOT\boolean{glshyperfirst}}{%
1472     \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
1473   }{%
1474     \@gls@link[#1]{#2}{\@glo@text}%
1475   }%
1476 }%

```

Indicate that this entry has now been used

```

1477 \glsunset{#2}}%
1478 }

```

`\Glspl` behaves in the same way as `\glspl`, except that the first letter of the link text is converted to uppercase (as with `\Gls`, if the first letter has an accent, it will need to be grouped).

`\Glspl`

```

1479 \newcommand*{\Glspl}{\@ifstar\@sGlspl\@Glspl}

```

Define the starred form:

```

1480 \newcommand*{\@sGlspl}[1][\@Glspl[hyper=false,#1]]

```

Defined the un-starred form. Need to determine if there is a final optional argument

```

1481 \newcommand*{\@Glspl}[2][\@Glspl@{#1}{#2}]{\@Glspl@{#1}{#2}[]}
1482 \new@ifnextchar[{\@Glspl@{#1}{#2}}{\@Glspl@{#1}{#2}[]}]

```

`\@Glspl@` Read in the final optional argument:

```

1483 \def\@Glspl@#1#2[#3]{%
1484   \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
    Save options in \@gls@link@opts and label in \@gls@link@label
1485   \def\@gls@link@opts{#1}%
1486   \def\@gls@link@label{#2}%
1487   \def\glslabel{#2}%

```

Determine what the link text should be (this is stored in `\@glo@text`). This needs to be expanded so that the `\@glo@text` can be passed to `\xmakefirstuc`.

```

1488 \ifglsused{#2}%
1489 {%
1490   \protected@edef\@glo@text{%
1491     \csname gls@\@glo@type @display\endcsname
1492     {\glsentryplural{#2}}{\glsentrydescplural{#2}}%
1493     {\glsentrysymbolplural{#2}}{#3}}%
1494 }%
1495 {%
1496   \protected@edef\@glo@text{%
1497     \csname gls@\@glo@type @displayfirst\endcsname
1498     {\glsentryfirstplural{#2}}{\glsentrydescplural{#2}}%
1499     {\glsentrysymbolplural{#2}}{#3}}%
1500 }%

```

Call `\@gls@link`. If footnote package option has been used and the glossary type is `\acronymtype`, suppress hyperlink for first use. Likewise if the `hyperfirst=false` package option is used.

```

1501 \ifglsused{#2}{%
1502   \@gls@link[#1]{#2}{%
1503     \expandafter\xmakefirstuc\expandafter{\@glo@text}}%
1504 }{%
1505   \gls@checkisacronymlist\@glo@type
1506   \ifthenelse{(\boolean{@glsisacronymlist})\AND
1507     \boolean{glsacrfootnote}) \OR \NOT\boolean{glshyperfirst}}{%
1508     \@gls@link[#1,hyper=false]{#2}{%
1509       \expandafter\xmakefirstuc\expandafter{\@glo@text}}%
1510   }{%
1511     \@gls@link[#1]{#2}{%
1512       \expandafter\xmakefirstuc\expandafter{\@glo@text}}%
1513   }%
1514 }%

```

Indicate that this entry has now been used

```

1515 \glsunset{#2}}%
1516 }

```

`\GLSp1` behaves like `\glspl` except that all the link text is converted to uppercase.

`\GLSp1`

```

1517 \newcommand*{\GLSp1}{\@ifstar\@sGLSp1\@GLSp1}

```

Define the starred form:

```

1518 \newcommand*{\@sGLSp1}[1] [] {\@GLSp1[hyper=false,#1]}

```

Defined the un-starred form. Need to determine if there is a final optional argument

```

1519 \newcommand*{\@GLSp1}[2] [] {%
1520 \new@ifnextchar[{\@GLSp1@{#1}{#2}}{\@GLSp1@{#1}{#2} []}]

```


\@GLSp1 Read in the final optional argument:

```

1521 \def\@GLSp1@#1#2[#3]{%
1522 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}%
    Save options in \@gls@link@opts and label in \@gls@link@label
1523 \def\@gls@link@opts{#1}%
1524 \def\@gls@link@label{#2}%
    Determine what the link text should be (this is stored in \@glo@text)
1525 \ifglsused{#2}{\def\@glo@text{%
1526 \csname gls@\@glo@type @display\endcsname
1527 {\glsentryplural{#2}}{\glsentrydescplural{#2}}{%
1528 \glsentrysymbolplural{#2}}{#3}}}%
1529 \def\@glo@text{%
1530 \csname gls@\@glo@type @displayfirst\endcsname
1531 {\glsentryfirstplural{#2}}{\glsentrydescplural{#2}}{%
1532 \glsentrysymbolplural{#2}}{#3}}}%

```

Call \@gls@link. If footnote package option has been used and the glossary type is \acronymtype, suppress hyperlink for first use. Likewise if the hyperfirst=false package option is used.

```

1533 \ifglsused{#2}{%
1534   \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
1535 }{%
1536   \gls@checkisacronymlist\@glo@type
1537   \ifthenelse{(\boolean{@glsisacronymlist})\AND
1538     \boolean{glsacrfootnote}) \OR \NOT\boolean{glshyperfirst}}{%
1539     \@gls@link[#1,hyper=false]{#2}{\MakeUppercase{\@glo@text}}%
1540   }{%
1541     \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
1542   }%
1543 }%

```

Indicate that this entry has now been used

```

1544 \glsunset{#2}}%
1545 }

```

\glsdisp \glsdisp[*options*]{*label*}{*text*} This is like \gls except that the link text is provided. This differs from \glslink in that it uses \glsdisplay or \glsdisplayfirst and unsets the first use flag.

First determine if we are using the starred form:

```

1546 \newcommand*{\glsdisp}{\ifstar\@sglsdisp\@glsdisp}

```

Define the starred form:

\@sgls

```

1547 \newcommand*{\@sglsdisp}[1][\@glsdisp[hyper=false,#1]]

```

Defined the un-starred form.

```

\@glsdisp
1548 \newcommand*{\@glsdisp}[3][]{%
1549   \glsdoifexists{#2}{%

1550     \edef\@glo@type{\glsentrytype{#2}}%

    Save options in \@gls@link@opts and label in \@gls@link@label
1551     \def\@gls@link@opts{#1}%
1552     \def\@gls@link@label{#2}%

    Determine what the link text should be (this is stored in \@glo@text)
1553     \ifglsused{#2}%
1554     {%
1555       \def\@glo@text{%
1556         \csname gls@\@glo@type @display\endcsname
1557         {#3}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{}}%
1558     }%
1559     {%
1560       \def\@glo@text{%
1561         \csname gls@\@glo@type @displayfirst\endcsname
1562         {#3}{\glsentrydesc{#2}}{\glsentrysymbol{#2}}{}}%
1563     }%

    Call \@gls@link. If footnote package option has been used and the glossary type
    is \acronymtype, suppress hyperlink for first use. Likewise if the hyperfirst=false
    package option is used.
1564     \ifglsused{#2}%
1565     {%
1566       \@gls@link[#1]{#2}{\@glo@text}%
1567     }%
1568     {%
1569       \gls@checkisacronymlist\@glo@type
1570       \ifthenelse{(\boolean{glsisacronymlist}\AND
1571         \boolean{glsacrfootnote})\OR \NOT\boolean{gls hyperfirst}}%
1572       {%
1573         \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
1574       }%
1575       {%
1576         \@gls@link[#1]{#2}{\@glo@text}%
1577       }%
1578     }%

    Indicate that this entry has now been used
1579     \glsunset{#2}%
1580   }%
1581 }

\glstext
1582 \newcommand*{\glstext}{\@ifstar\@sglstext\@glstext}

```

`\glstext` behaves like `\gls` except it always uses the value given by the text key and it doesn't mark the entry as used.

Define the starred form:

```
1583 \newcommand*{\@sglstext}[1][\@glstext[hyper=false,#1]]
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1584 \newcommand*{\@glstext}[2][\%
```

```
1585 \new@ifnextchar[\@glstext@{#1}{#2}]{\@glstext@{#1}{#2}[]}]
```

Read in the final optional argument:

```
1586 \def\@glstext@#1#2[#3]{\%
```

```
1587 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1588 \protected@edef\@glo@text{\glsentrytext{#2}}%
```

Call \@gls@link

```
1589 \@gls@link[#1]{#2}{\@glo@text#3}%
```

```
1590 }%
```

```
1591 }
```

\GLStext behaves like \glstext except the text is converted to uppercase.

\GLStext

```
1592 \newcommand*{\GLStext}{\@ifstar\@sGLStext\@GLStext}
```

Define the starred form:

```
1593 \newcommand*{\@sGLStext}[1][\@GLStext[hyper=false,#1]]
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1594 \newcommand*{\@GLStext}[2][\%
```

```
1595 \new@ifnextchar[\@GLStext@{#1}{#2}]{\@GLStext@{#1}{#2}[]}]
```

Read in the final optional argument:

```
1596 \def\@GLStext@#1#2[#3]{\%
```

```
1597 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1598 \protected@edef\@glo@text{\glsentrytext{#2}}%
```

Call \@gls@link

```
1599 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
```

```
1600 }%
```

```
1601 }
```

\Glstext behaves like \glstext except that the first letter of the text is converted to uppercase.

\Glstext

```
1602 \newcommand*{\Glstext}{\@ifstar\@sGlstext\@Glstext}
```

Define the starred form:

```
1603 \newcommand*{\@sGlstext}[1][\@Glstext[hyper=false,#1]]
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1604 \newcommand*{\@Glstext}[2] [] {%
1605 \new@ifnextchar [{\@Glstext@{#1}{#2}}{\@Glstext@{#1}{#2} []}}
```

Read in the final optional argument:

```
1606 \def\@Glstext@#1#2[#3] {%
1607 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1608 \protected@edef\@glo@text{\glsentrytext{#2}}%
```

Call \@gls@link

```
1609 \@gls@link[#1]{#2}{%
1610 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
1611 }%
1612 }
```

\glsfirst behaves like \gls except it always uses the value given by the first key and it doesn't mark the entry as used.

\glsfirst

```
1613 \newcommand*{\glsfirst}{\@ifstar\@sglsfirst\@glsfirst}
```

Define the starred form:

```
1614 \newcommand*{\@sglsfirst}[1] [] {\@glsfirst[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1615 \newcommand*{\@glsfirst}[2] [] {%
1616 \new@ifnextchar [{\@glsfirst@{#1}{#2}}{\@glsfirst@{#1}{#2} []}}
```

Read in the final optional argument:

```
1617 \def\@glsfirst@#1#2[#3] {%
1618 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1619 \protected@edef\@glo@text{\glsentryfirst{#2}}%
```

Call \@gls@link

```
1620 \@gls@link[#1]{#2}{\@glo@text#3}%
1621 }%
1622 }
```

\Glsfirst behaves like \glsfirst except it displays the first letter in uppercase.

\Glsfirst

```
1623 \newcommand*{\Glsfirst}{\@ifstar\@sGlsfirst\@Glsfirst}
```

Define the starred form:

```
1624 \newcommand*{\@sGlsfirst}[1] [] {\@Glsfirst[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1625 \newcommand*{\Glsfirst}[2] [] {%
1626 \new@ifnextchar [{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2} []}}
```

Read in the final optional argument:

```
1627 \def\@Glsfirst@#1#2[#3] {%
1628 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1629 \protected@edef\@glo@text{\glsentryfirst{#2}}%
```

Call \@gls@link

```
1630 \@gls@link[#1]{#2}{%
1631 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
1632 }%
1633 }
```

\Glsfirst behaves like \Glsfirst except it displays the text in uppercase.

\Glsfirst

```
1634 \newcommand*{\Glsfirst}{\@ifstar\@sGlsfirst\@Glsfirst}
```

Define the starred form:

```
1635 \newcommand*{\@sGlsfirst}[1] [] {\@Glsfirst[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1636 \newcommand*{\@Glsfirst}[2] [] {%
1637 \new@ifnextchar [{\@Glsfirst@{#1}{#2}}{\@Glsfirst@{#1}{#2} []}}
```

Read in the final optional argument:

```
1638 \def\@Glsfirst@#1#2[#3] {%
1639 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1640 \protected@edef\@glo@text{\glsentryfirst{#2}}%
```

Call \@gls@link

```
1641 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
1642 }%
1643 }
```

\glsplural behaves like \gls except it always uses the value given by the plural key and it doesn't mark the entry as used.

\glsplural

```
1644 \newcommand*{\glsplural}{\@ifstar\@sglsplural\@glsplural}
```

Define the starred form:

```
1645 \newcommand*{\@sglsplural}[1] [] {\@glsplural[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1646 \newcommand*{\glsplural}[2] [] {%
1647 \new@ifnextchar[{\@glsplural@{#1}{#2}}{\@glsplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
1648 \def\@glsplural@#1#2[#3] {%
1649 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1650 \protected@edef\@glo@text{\glsentryplural{#2}}%
```

Call \@gls@link

```
1651 \@gls@link[#1]{#2}{\@glo@text#3}%
1652 }%
1653 }
```

\Glsplural behaves like \glsplural except that the first letter is converted to uppercase.

\Glsplural

```
1654 \newcommand*{\Glsplural}{\@ifstar\@sGlsplural\@Glsplural}
```

Define the starred form:

```
1655 \newcommand*{\@sGlsplural}[1] [] {\@Glsplural[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1656 \newcommand*{\@Glsplural}[2] [] {%
1657 \new@ifnextchar[{\@Glsplural@{#1}{#2}}{\@Glsplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
1658 \def\@Glsplural@#1#2[#3] {%
1659 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1660 \protected@edef\@glo@text{\glsentryplural{#2}}%
```

Call \@gls@link

```
1661 \@gls@link[#1]{#2}{%
1662   \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
1663 }%
1664 }
```

\GLSplural behaves like \glsplural except that the text is converted to uppercase.

\GLSplural

```
1665 \newcommand*{\GLSplural}{\@ifstar\@sGLSplural\@GLSplural}
```

Define the starred form:

```
1666 \newcommand*{\@sGLSplural}[1] [] {\@GLSplural[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1667 \newcommand*{\@GLSplural}[2] [] {%
1668 \new@ifnextchar[{\@GLSplural@{#1}{#2}}{\@GLSplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
1669 \def\@GLSplural@#1#2[#3] {%
1670 \glsdoifexists{#2}{\edef\@glo@type{\glstrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1671 \protected@edef\@glo@text{\glstryplural{#2}}%
```

Call \@gls@link

```
1672 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
1673 }%
1674 }
```

\glsfirstplural behaves like \gls except it always uses the value given by the firstplural key and it doesn't mark the entry as used.

\glsfirstplural

```
1675 \newcommand*{\glsfirstplural}{\@ifstar\sglsfirstplural\@glsfirstplural}
```

Define the starred form:

```
1676 \newcommand*{\sglsfirstplural}[1] [] {\@glsfirstplural[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1677 \newcommand*{\@glsfirstplural}[2] [] {%
1678 \new@ifnextchar[{\@glsfirstplural@{#1}{#2}}{\@glsfirstplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
1679 \def\@glsfirstplural@#1#2[#3] {%
1680 \glsdoifexists{#2}{\edef\@glo@type{\glstrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1681 \protected@edef\@glo@text{\glstryfirstplural{#2}}%
```

Call \@gls@link

```
1682 \@gls@link[#1]{#2}{\@glo@text#3}%
1683 }%
1684 }
```

\Glsfirstplural behaves like \glsfirstplural except that the first letter is converted to uppercase.

\Glsfirstplural

```
1685 \newcommand*{\Glsfirstplural}{\@ifstar\@sGlsfirstplural\@Glsfirstplural}
```

Define the starred form:

```
1686 \newcommand*{\@sGlsfirstplural}[1] [] {\@Glsfirstplural[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1687 \newcommand*{\Glsfirstplural}[2] [] {%
1688 \new@ifnextchar[{\@Glsfirstplural@{#1}{#2}}{\@Glsfirstplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
1689 \def\@Glsfirstplural@#1#2[#3] {%
1690 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1691 \protected@edef\@glo@text{\glsentryfirstplural{#2}}%
```

Call \@gls@link

```
1692 \@gls@link[#1]{#2}{%
1693 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
1694 }%
1695 }
```

\Glsfirstplural behaves like \glsfirstplural except that the link text is converted to uppercase.

\GLSfirstplural

```
1696 \newcommand*{\GLSfirstplural}{\@ifstar\@sGLSfirstplural\@GLSfirstplural}
```

Define the starred form:

```
1697 \newcommand*{\@sGLSfirstplural}[1] [] {\@GLSfirstplural[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1698 \newcommand*{\@GLSfirstplural}[2] [] {%
1699 \new@ifnextchar[{\@GLSfirstplural@{#1}{#2}}{\@GLSfirstplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
1700 \def\@GLSfirstplural@#1#2[#3] {%
1701 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1702 \protected@edef\@glo@text{\glsentryfirstplural{#2}}%
```

Call \@gls@link

```
1703 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
1704 }%
1705 }
```

\glsname behaves like \gls except it always uses the value given by the name key and it doesn't mark the entry as used.

\glsname

```
1706 \newcommand*{\glsname}{\@ifstar\@sglsname\@glsname}
```

Define the starred form:

```
1707 \newcommand*{\@sglsname}[1] [] {\@glsname[hyper=false,#1]}
```


Defined the un-starred form. Need to determine if there is a final optional argument

```
1708 \newcommand*{\@glsname}[2][]{%
1709 \new@ifnextchar[{\@glsname@{#1}{#2}}{\@glsname@{#1}{#2}[]}}
```

Read in the final optional argument:

```
1710 \def\@glsname@#1#2[#3]{%
1711 \glsdoifexists{#2}{\edef\@glo@type{\glstrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1712 \protected@edef\@glo@text{\glstryname{#2}}%
```

Call \@gls@link

```
1713 \@gls@link[#1]{#2}{\@glo@text#3}%
1714 }%
1715 }
```

\Glsname behaves like \glsname except that the first letter is converted to uppercase.

\Glsname

```
1716 \newcommand*{\Glsname}{\@ifstar\@sGlsname\@Glsname}
```

Define the starred form:

```
1717 \newcommand*{\@sGlsname}[1][]{\@Glsname[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1718 \newcommand*{\@Glsname}[2][]{%
1719 \new@ifnextchar[{\@Glsname@{#1}{#2}}{\@Glsname@{#1}{#2}[]}}
```

Read in the final optional argument:

```
1720 \def\@Glsname@#1#2[#3]{%
1721 \glsdoifexists{#2}{\edef\@glo@type{\glstrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1722 \protected@edef\@glo@text{\glstryname{#2}}%
```

Call \@gls@link

```
1723 \@gls@link[#1]{#2}{%
1724 \expandafter\makefirstuc\expandafter{\@glo@text#3}%
1725 }%
1726 }
```

\GLSname behaves like \glsname except that the link text is converted to uppercase.

\GLSname

```
1727 \newcommand*{\GLSname}{\@ifstar\@sGLSname\@GLSname}
```

Define the starred form:

```
1728 \newcommand*{\@sGLSname}[1][]{\@GLSname[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1729 \newcommand*{\@GLSname}[2] [] {%
1730 \new@ifnextchar[{\@GLSname@{#1}{#2}}{\@GLSname@{#1}{#2} []}}
```

Read in the final optional argument:

```
1731 \def\@GLSname@#1#2[#3] {%
1732 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1733 \protected@edef\@glo@text{\glsentryname{#2}}%
```

Call \@gls@link

```
1734 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
1735 }%
1736 }
```

\glsdesc behaves like \gls except it always uses the value given by the description key and it doesn't mark the entry as used.

\glsdesc

```
1737 \newcommand*{\glsdesc}{\@ifstar\@sglsdesc\@glsdesc}
```

Define the starred form:

```
1738 \newcommand*{\@sglsdesc}[1] [] {\@glsdesc[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1739 \newcommand*{\@glsdesc}[2] [] {%
1740 \new@ifnextchar[{\@glsdesc@{#1}{#2}}{\@glsdesc@{#1}{#2} []}}
```

Read in the final optional argument:

```
1741 \def\@glsdesc@#1#2[#3] {%
1742 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1743 \protected@edef\@glo@text{\glsentrydesc{#2}}%
```

Call \@gls@link

```
1744 \@gls@link[#1]{#2}{\@glo@text#3}%
1745 }%
1746 }
```

\Glsdesc behaves like \glsdesc except that the first letter is converted to uppercase.

\Glsdesc

```
1747 \newcommand*{\Glsdesc}{\@ifstar\@sGlsdesc\@Glsdesc}
```

Define the starred form:

```
1748 \newcommand*{\@sGlsdesc}[1] [] {\@Glsdesc[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1749 \newcommand*{\@Glsdesc}[2] [] {%
1750 \new@ifnextchar [{\@Glsdesc@{#1}{#2}}{\@Glsdesc@{#1}{#2} []}}
```

Read in the final optional argument:

```
1751 \def\@Glsdesc@#1#2[#3] {%
1752 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
1753 \protected@edef\@glo@text{\glsentrydesc{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1753 \protected@edef\@glo@text{\glsentrydesc{#2}}%
1754 \gls@link
1755 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
1756 }%
1757 }
```

\GLSdesc behaves like \glsdesc except that the link text is converted to uppercase.

\GLSdesc

```
1758 \newcommand*{\GLSdesc}{\@ifstar\@sGLSdesc\@GLSdesc}
```

Define the starred form:

```
1759 \newcommand*{\@sGLSdesc}[1] [] {\@GLSdesc[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1760 \newcommand*{\@GLSdesc}[2] [] {%
1761 \new@ifnextchar [{\@GLSdesc@{#1}{#2}}{\@GLSdesc@{#1}{#2} []}}
```

Read in the final optional argument:

```
1762 \def\@GLSdesc@#1#2[#3] {%
1763 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
1764 \protected@edef\@glo@text{\glsentrydesc{#2}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1764 \protected@edef\@glo@text{\glsentrydesc{#2}}%
1765 \gls@link
1766 \expandafter\MakeUppercase\expandafter{\@glo@text}#3}%
1767 }
```

\glsdescplural behaves like \gls except it always uses the value given by the descriptionplural key and it doesn't mark the entry as used.

\glsdescplural

```
1768 \newcommand*{\glsdescplural}{\@ifstar\@sglsdescplural\@glsdescplural}
```

Define the starred form:

```
1769 \newcommand*{\@sglsdescplural}[1] [] {\@glsdescplural[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1770 \newcommand*{\glsdescplural}[2] [] {%
1771 \new@ifnextchar[{\@glsdescplural@{#1}{#2}}{\@glsdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
1772 \def\@glsdescplural@#1#2[#3]{%
1773 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1774 \protected@edef\@glo@text{\glsentrydescplural{#2}}%
```

Call \@gls@link

```
1775 \@gls@link[#1]{#2}{\@glo@text#3}%
1776 }%
1777 }
```

\Glsdescplural behaves like \glsdescplural except that the first letter is converted to uppercase.

\Glsdescplural

```
1778 \newcommand*{\Glsdescplural}{\@ifstar\@sGlsdescplural\@Glsdescplural}
```

Define the starred form:

```
1779 \newcommand*{\@sGlsdescplural}[1] [] {\@Glsdescplural[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1780 \newcommand*{\@Glsdescplural}[2] [] {%
1781 \new@ifnextchar[{\@Glsdescplural@{#1}{#2}}{\@Glsdescplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
1782 \def\@Glsdescplural@#1#2[#3]{%
1783 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1784 \protected@edef\@glo@text{\glsentrydescplural{#2}}%
```

Call \@gls@link

```
1785 \@gls@link[#1]{#2}{%
1786 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
1787 }%
1788 }
```

\GLSdescplural behaves like \glsdescplural except that the link text is converted to uppercase.

\GLSdescplural

```
1789 \newcommand*{\GLSdescplural}{\@ifstar\@sGLSdescplural\@GLSdescplural}
```

Define the starred form:

```
1790 \newcommand*{\@sGLSdescplural}[1] [] {\@GLSdescplural[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1791 \newcommand*{\GLSdescplural}[2] [] {%
1792 \new@ifnextchar[{\@GLSdescplural@{#1}{#2}}{\@GLSdescplural@{#1}{#2} []}}
```

Read in the final optional argument:

```
1793 \def\@GLSdescplural@#1#2[#3] {%
1794 \glsoifexists{#2}{\edef\@glo@type{\glstrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1795 \protected@edef\@glo@text{\glstrydescplural{#2}}%
```

Call \@gls@link

```
1796 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
1797 }%
1798 }
```

\glsymbol behaves like \gls except it always uses the value given by the symbol key and it doesn't mark the entry as used.

\glsymbol

```
1799 \newcommand*{\glsymbol}{\@ifstar\@sglsymbol\@glsymbol}
```

Define the starred form:

```
1800 \newcommand*{\@sglsymbol}[1] [] {\@glsymbol[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1801 \newcommand*{\@glsymbol}[2] [] {%
1802 \new@ifnextchar[{\@glsymbol@{#1}{#2}}{\@glsymbol@{#1}{#2} []}}
```

Read in the final optional argument:

```
1803 \def\@glsymbol@#1#2[#3] {%
1804 \glsoifexists{#2}{\edef\@glo@type{\glstrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1805 \protected@edef\@glo@text{\glstrysymbol{#2}}%
```

Call \@gls@link

```
1806 \@gls@link[#1]{#2}{\@glo@text#3}%
1807 }%
1808 }
```

\Glsymbol behaves like \glsymbol except that the first letter is converted to uppercase.

\Glsymbol

```
1809 \newcommand*{\Glsymbol}{\@ifstar\@sGlsymbol\@Glsymbol}
```

Define the starred form:

```
1810 \newcommand*{\@sGlsymbol}[1] [] {\@Glsymbol[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1811 \newcommand*{\@Glsymbol}[2] [] {%
1812 \new@ifnextchar [{\@Glsymbol@{#1}{#2}}{\@Glsymbol@{#1}{#2} []}}
```

Read in the final optional argument:

```
1813 \def\@Glsymbol@#1#2[#3] {%
1814 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1815 \protected@edef\@glo@text{\glsentrysymbol{#2}}%
```

Call \@gls@link

```
1816 \@gls@link[#1]{#2}{%
1817 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
1818 }%
1819 }
```

\GLSsymbol behaves like \glsymbol except that the link text is converted to uppercase.

\GLSsymbol

```
1820 \newcommand*{\GLSsymbol}{\@ifstar\@sGLSsymbol\@GLSsymbol}
```

Define the starred form:

```
1821 \newcommand*{\@sGLSsymbol}[1] [] {\@GLSsymbol[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1822 \newcommand*{\@GLSsymbol}[2] [] {%
1823 \new@ifnextchar [{\@GLSsymbol@{#1}{#2}}{\@GLSsymbol@{#1}{#2} []}}
```

Read in the final optional argument:

```
1824 \def\@GLSsymbol@#1#2[#3] {%
1825 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1826 \protected@edef\@glo@text{\glsentrysymbol{#2}}%
```

Call \@gls@link

```
1827 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}#3}%
1828 }%
1829 }
```

\glsymbolplural behaves like \gls except it always uses the value given by the symbolplural key and it doesn't mark the entry as used.

\glsymbolplural

```
1830 \newcommand*{\glsymbolplural}{\@ifstar\@sglsymbolplural\@glsymbolplural}
```

Define the starred form:

```
1831 \newcommand*{\@sglsymbolplural}[1] [] {\@glsymbolplural[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1832 \newcommand*{\glssymbolplural}[2] [] {%
1833 \new@ifnextchar[{\@glssymbolplural@{#1}{#2}}{\@glssymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
1834 \def\@glssymbolplural@#1#2[#3] {%
1835 \glsoifexists{#2}{\edef\@glo@type{\glstrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1836 \protected@edef\@glo@text{\glstrysymbolplural{#2}}%
```

Call \@gls@link

```
1837 \@gls@link[#1]{#2}{\@glo@text#3}%
1838 }%
1839 }
```

\Glsymbolplural behaves like \glssymbolplural except that the first letter is converted to uppercase.

\Glsymbolplural

```
1840 \newcommand*{\Glsymbolplural}{\@ifstar\@sGlsymbolplural\@Glsymbolplural}
```

Define the starred form:

```
1841 \newcommand*{\@sGlsymbolplural}[1] [] {\@Glsymbolplural[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1842 \newcommand*{\@Glsymbolplural}[2] [] {%
1843 \new@ifnextchar[{\@Glsymbolplural@{#1}{#2}}{\@Glsymbolplural@{#1}{#2} []}]}
```

Read in the final optional argument:

```
1844 \def\@Glsymbolplural@#1#2[#3] {%
1845 \glsoifexists{#2}{\edef\@glo@type{\glstrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1846 \protected@edef\@glo@text{\glstrysymbolplural{#2}}%
```

Call \@gls@link

```
1847 \@gls@link[#1]{#2}{%
1848   \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
1849 }%
1850 }
```

\GLSsymbolplural behaves like \glssymbolplural except that the link text is converted to uppercase.

\GLSsymbolplural

```
1851 \newcommand*{\GLSsymbolplural}{\@ifstar\@sGLSsymbolplural\@GLSsymbolplural}
```

Define the starred form:

```
1852 \newcommand*{\@sGLSsymbolplural}[1] [] {\@GLSsymbolplural[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```

1853 \newcommand*{\GLSsymbolplural}[2] [] {%
1854 \new@ifnextchar [{\@GLSsymbolplural@{#1}{#2}}{\@GLSsymbolplural@{#1}{#2} [] }}
    Read in the final optional argument:
1855 \def\@GLSsymbolplural@#1#2[#3] {%
1856 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
    Determine what the link text should be (this is stored in \@glo@text)
1857 \protected@edef\@glo@text{\glsentrysymbolplural{#2}}%
    Call \@gls@link
1858 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
1859 }%
1860 }

```

`\glsuseri` behaves like `\gls` except it always uses the value given by the `user1` key and it doesn't mark the entry as used.

`\glsuseri`

```

1861 \newcommand*{\glsuseri}{\@ifstar\sglsuseri\@glsuseri}
    Define the starred form:
1862 \newcommand*{\sglsuseri}[1] [] {\@glsuseri[hyper=false,#1]}
    Defined the un-starred form. Need to determine if there is a final optional argument
1863 \newcommand*{\@glsuseri}[2] [] {%
1864 \new@ifnextchar [{\@glsuseri@{#1}{#2}}{\@glsuseri@{#1}{#2} [] }}
    Read in the final optional argument:
1865 \def\@glsuseri@#1#2[#3] {%
1866 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
    Determine what the link text should be (this is stored in \@glo@text)
1867 \protected@edef\@glo@text{\glsentryuseri{#2}}%
    Call \@gls@link
1868 \@gls@link[#1]{#2}{\@glo@text#3}%
1869 }%
1870 }

```

`\Glsuseri` behaves like `\glsuseri` except that the first letter is converted to uppercase.

`\Glsuseri`

```

1871 \newcommand*{\Glsuseri}{\@ifstar\sglsuseri\@Glsuseri}
    Define the starred form:
1872 \newcommand*{\sglsuseri}[1] [] {\@Glsuseri[hyper=false,#1]}

```


Defined the un-starred form. Need to determine if there is a final optional argument

```
1873 \newcommand*{\@Glsuseri}[2] [] {%
1874 \new@ifnextchar [{\@Glsuseri@{#1}{#2}}{\@Glsuseri@{#1}{#2} []}}
```

Read in the final optional argument:

```
1875 \def\@Glsuseri@#1#2[#3] {%
1876 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1877 \protected@edef\@glo@text{\glsentryuseri{#2}}%
```

Call \@gls@link

```
1878 \@gls@link[#1]{#2}{%
1879 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
1880 }%
1881 }
```

\GLSuseri behaves like \glsuseri except that the link text is converted to uppercase.

\GLSuseri

```
1882 \newcommand*{\GLSuseri}{\@ifstar\@sGLSuseri\@GLSuseri}
```

Define the starred form:

```
1883 \newcommand*{\@sGLSuseri}[1] [] {\@GLSuseri[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1884 \newcommand*{\@GLSuseri}[2] [] {%
1885 \new@ifnextchar [{\@GLSuseri@{#1}{#2}}{\@GLSuseri@{#1}{#2} []}}
```

Read in the final optional argument:

```
1886 \def\@GLSuseri@#1#2[#3] {%
1887 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1888 \protected@edef\@glo@text{\glsentryuseri{#2}}%
```

Call \@gls@link

```
1889 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}#3}%
1890 }%
1891 }
```

\glsuserii behaves like \gls except it always uses the value given by the user2 key and it doesn't mark the entry as used.

\glsuserii

```
1892 \newcommand*{\glsuserii}{\@ifstar\@sglsuserii\@glsuserii}
```

Define the starred form:

```
1893 \newcommand*{\@sglsuserii}[1] [] {\@glsuserii[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1894 \newcommand*{\@glsuserii}[2] [] {%
1895 \new@ifnextchar[{\@glsuserii@{#1}{#2}}{\@glsuserii@{#1}{#2} []}}
```

Read in the final optional argument:

```
1896 \def\@glsuserii@#1#2[#3] {%
1897 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1898 \protected@edef\@glo@text{\glsentryuserii{#2}}%
```

Call \@gls@link

```
1899 \@gls@link[#1]{#2}{\@glo@text#3}%
1900 }%
1901 }
```

\Glsuserii behaves like \glsuserii except that the first letter is converted to uppercase.

\Glsuserii

```
1902 \newcommand*{\Glsuserii}{\@ifstar\@sGlsuserii\@Glsuserii}
```

Define the starred form:

```
1903 \newcommand*{\@sGlsuserii}[1] [] {\@Glsuserii[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1904 \newcommand*{\@Glsuserii}[2] [] {%
1905 \new@ifnextchar[{\@Glsuserii@{#1}{#2}}{\@Glsuserii@{#1}{#2} []}}
```

Read in the final optional argument:

```
1906 \def\@Glsuserii@#1#2[#3] {%
1907 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1908 \protected@edef\@glo@text{\glsentryuserii{#2}}%
```

Call \@gls@link

```
1909 \@gls@link[#1]{#2}{%
1910 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
1911 }%
1912 }
```

\GLSuserii behaves like \glsuserii except that the link text is converted to uppercase.

\GLSuserii

```
1913 \newcommand*{\GLSuserii}{\@ifstar\@sGLSuserii\@GLSuserii}
```

Define the starred form:

```
1914 \newcommand*{\@sGLSuserii}[1] [] {\@GLSuserii[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1915 \newcommand*{\GLSuserii}[2] [] {%
1916 \new@ifnextchar[{\@GLSuserii@{#1}{#2}}{\@GLSuserii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
1917 \def\@GLSuserii@#1#2[#3] {%
1918 \glsdoifexists{#2}{\edef\@glo@type{\glstrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1919 \protected@edef\@glo@text{\glstryuserii{#2}}%
```

Call \@gls@link

```
1920 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
1921 }%
1922 }
```

\glsuseriii behaves like \gls except it always uses the value given by the user3 key and it doesn't mark the entry as used.

\glsuseriii

```
1923 \newcommand*{\glsuseriii}{\@ifstar\sglsuseriii\@glsuseriii}
```

Define the starred form:

```
1924 \newcommand*{\sglsuseriii}[1] [] {\@glsuseriii[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1925 \newcommand*{\@glsuseriii}[2] [] {%
1926 \new@ifnextchar[{\@glsuseriii@{#1}{#2}}{\@glsuseriii@{#1}{#2} []}]}
```

Read in the final optional argument:

```
1927 \def\@glsuseriii@#1#2[#3] {%
1928 \glsdoifexists{#2}{\edef\@glo@type{\glstrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1929 \protected@edef\@glo@text{\glstryuseriii{#2}}%
```

Call \@gls@link

```
1930 \@gls@link[#1]{#2}{\@glo@text#3}}%
1931 }%
1932 }
```

\Glsuseriii behaves like \glsuseriii except that the first letter is converted to uppercase.

\Glsuseriii

```
1933 \newcommand*{\Glsuseriii}{\@ifstar\sglsuseriii\@Glsuseriii}
```

Define the starred form:

```
1934 \newcommand*{\sglsuseriii}[1] [] {\@Glsuseriii[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1935 \newcommand*{\@Glsuseriii}[2] [] {%
1936 \new@ifnextchar [{\@Glsuseriii@{#1}{#2}}{\@Glsuseriii@{#1}{#2} []}}
```

Read in the final optional argument:

```
1937 \def\@Glsuseriii@#1#2[#3] {%
1938 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1939 \protected@edef\@glo@text{\glsentryuseriii{#2}}%
```

Call \@gls@link

```
1940 \@gls@link[#1]{#2}{%
1941 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
1942 }%
1943 }
```

\GLSuseriii behaves like \glsuseriii except that the link text is converted to uppercase.

\GLSuseriii

```
1944 \newcommand*{\GLSuseriii}{\@ifstar\@sGLSuseriii\@GLSuseriii}
```

Define the starred form:

```
1945 \newcommand*{\@sGLSuseriii}[1] [] {\@GLSuseriii[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1946 \newcommand*{\@GLSuseriii}[2] [] {%
1947 \new@ifnextchar [{\@GLSuseriii@{#1}{#2}}{\@GLSuseriii@{#1}{#2} []}}
```

Read in the final optional argument:

```
1948 \def\@GLSuseriii@#1#2[#3] {%
1949 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1950 \protected@edef\@glo@text{\glsentryuseriii{#2}}%
```

Call \@gls@link

```
1951 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}#3}%
1952 }%
1953 }
```

\glsuseriv behaves like \gls except it always uses the value given by the user4 key and it doesn't mark the entry as used.

\glsuseriv

```
1954 \newcommand*{\glsuseriv}{\@ifstar\@sglsuseriv\@glsuseriv}
```

Define the starred form:

```
1955 \newcommand*{\@sglsuseriv}[1] [] {\@glsuseriv[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1956 \newcommand*{\@glsuseriv}[2] [] {%
1957 \new@ifnextchar[{\@glsuseriv@{#1}{#2}}{\@glsuseriv@{#1}{#2} []}}
```

Read in the final optional argument:

```
1958 \def\@glsuseriv@#1#2[#3] {%
1959 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1960 \protected@edef\@glo@text{\glsentryuseriv{#2}}%
```

Call \@gls@link

```
1961 \@gls@link[#1]{#2}{\@glo@text#3}%
1962 }%
1963 }
```

\Glsuseriv behaves like \glsuseriv except that the first letter is converted to uppercase.

\Glsuseriv

```
1964 \newcommand*{\Glsuseriv}{\@ifstar\@sGlsuseriv\@Glsuseriv}
```

Define the starred form:

```
1965 \newcommand*{\@sGlsuseriv}[1] [] {\@Glsuseriv[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1966 \newcommand*{\@Glsuseriv}[2] [] {%
1967 \new@ifnextchar[{\@Glsuseriv@{#1}{#2}}{\@Glsuseriv@{#1}{#2} []}}
```

Read in the final optional argument:

```
1968 \def\@Glsuseriv@#1#2[#3] {%
1969 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1970 \protected@edef\@glo@text{\glsentryuseriv{#2}}%
```

Call \@gls@link

```
1971 \@gls@link[#1]{#2}{%
1972 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
1973 }%
1974 }
```

\GLSuseriv behaves like \glsuseriv except that the link text is converted to uppercase.

\GLSuseriv

```
1975 \newcommand*{\GLSuseriv}{\@ifstar\@sGLSuseriv\@GLSuseriv}
```

Define the starred form:

```
1976 \newcommand*{\@sGLSuseriv}[1] [] {\@GLSuseriv[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1977 \newcommand*{\@GLSuseriv}[2] [] {%
1978 \new@ifnextchar[{\@GLSuseriv@{#1}{#2}}{\@GLSuseriv@{#1}{#2} []}}
```

Read in the final optional argument:

```
1979 \def\@GLSuseriv@#1#2[#3] {%
1980 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1981 \protected@edef\@glo@text{\glsentryuseriv{#2}}%
```

Call \@gls@link

```
1982 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
1983 }%
1984 }
```

\glsuserv behaves like \gls except it always uses the value given by the user5 key and it doesn't mark the entry as used.

\glsuserv

```
1985 \newcommand*{\glsuserv}{\@ifstar\@sglsuserv\@glsuserv}
```

Define the starred form:

```
1986 \newcommand*{\@sglsuserv}[1] [] {\@glsuserv[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1987 \newcommand*{\@glsuserv}[2] [] {%
1988 \new@ifnextchar[{\@glsuserv@{#1}{#2}}{\@glsuserv@{#1}{#2} []}}
```

Read in the final optional argument:

```
1989 \def\@glsuserv@#1#2[#3] {%
1990 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
1991 \protected@edef\@glo@text{\glsentryuseriv{#2}}%
```

Call \@gls@link

```
1992 \@gls@link[#1]{#2}{\@glo@text#3}%
1993 }%
1994 }
```

\Glsuserv behaves like \glsuserv except that the first letter is converted to uppercase.

\Glsuserv

```
1995 \newcommand*{\Glsuserv}{\@ifstar\@sGlsuserv\@Glsuserv}
```

Define the starred form:

```
1996 \newcommand*{\@sGlsuserv}[1] [] {\@Glsuserv[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
1997 \newcommand*{\@Glsuserv}[2] [] {%
1998 \new@ifnextchar [{\@Glsuserv@{#1}{#2}}{\@Glsuserv@{#1}{#2} []}}
```

Read in the final optional argument:

```
1999 \def\@Glsuserv@#1#2[#3] {%
2000 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2001 \protected@edef\@glo@text{\glsentryuserv{#2}}%
```

Call \@gls@link

```
2002 \@gls@link[#1]{#2}{%
2003 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2004 }%
2005 }
```

\GLSuserv behaves like \glsuserv except that the link text is converted to uppercase.

\GLSuserv

```
2006 \newcommand*{\GLSuserv}{\@ifstar\@sGLSuserv\@GLSuserv}
```

Define the starred form:

```
2007 \newcommand*{\@sGLSuserv}[1] [] {\@GLSuserv[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
2008 \newcommand*{\@GLSuserv}[2] [] {%
2009 \new@ifnextchar [{\@GLSuserv@{#1}{#2}}{\@GLSuserv@{#1}{#2} []}}
```

Read in the final optional argument:

```
2010 \def\@GLSuserv@#1#2[#3] {%
2011 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2012 \protected@edef\@glo@text{\glsentryuserv{#2}}%
```

Call \@gls@link

```
2013 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2014 }%
2015 }
```

\glsuservi behaves like \gls except it always uses the value given by the user6 key and it doesn't mark the entry as used.

\glsuservi

```
2016 \newcommand*{\glsuservi}{\@ifstar\@sglsuservi\@glsuservi}
```

Define the starred form:

```
2017 \newcommand*{\@sglsuservi}[1] [] {\@glsuservi[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
2018 \newcommand*{\@glsuservi}[2] [] {%
2019 \new@ifnextchar[{\@glsuservi@{#1}{#2}}{\@glsuservi@{#1}{#2} []}]}
```

Read in the final optional argument:

```
2020 \def\@glsuservi@#1#2[#3] {%
2021 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2022 \protected@edef\@glo@text{\glsentryuservi{#2}}%
```

Call \@gls@link

```
2023 \@gls@link[#1]{#2}{\@glo@text#3}%
2024 }%
2025 }
```

\Glsuservi behaves like \glsuservi except that the first letter is converted to uppercase.

\Glsuservi

```
2026 \newcommand*{\Glsuservi}{\@ifstar\@sGlsuservi\@Glsuservi}
```

Define the starred form:

```
2027 \newcommand*{\@sGlsuservi}[1] [] {\@Glsuservi[hyper=false,#1]}
```

Defined the un-starred form. Need to determine if there is a final optional argument

```
2028 \newcommand*{\@Glsuservi}[2] [] {%
2029 \new@ifnextchar[{\@Glsuservi@{#1}{#2}}{\@Glsuservi@{#1}{#2} []}]}
```

Read in the final optional argument:

```
2030 \def\@Glsuservi@#1#2[#3] {%
2031 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
```

Determine what the link text should be (this is stored in \@glo@text)

```
2032 \protected@edef\@glo@text{\glsentryuservi{#2}}%
```

Call \@gls@link

```
2033 \@gls@link[#1]{#2}{%
2034 \expandafter\makefirstuc\expandafter{\@glo@text}#3}%
2035 }%
2036 }
```

\GLSuservi behaves like \glsuservi except that the link text is converted to uppercase.

\GLSuservi

```
2037 \newcommand*{\GLSuservi}{\@ifstar\@sGLSuservi\@GLSuservi}
```

Define the starred form:

```
2038 \newcommand*{\@sGLSuservi}[1] [] {\@GLSuservi[hyper=false,#1]}
```


Defined the un-starred form. Need to determine if there is a final optional argument

```

2039 \newcommand*{\@GLSuservi}[2][\%
2040 \new@ifnextchar[\@GLSuservi@{#1}{#2}]{\@GLSuservi@{#1}{#2}[]}}
    Read in the final optional argument:
2041 \def\@GLSuservi@#1#2[#3]{\%
2042 \glsdoifexists{#2}{\edef\@glo@type{\glsentrytype{#2}}}%
    Determine what the link text should be (this is stored in \@glo@text)
2043 \protected@edef\@glo@text{\glsentryuservi{#2}}%
    Call \@gls@link
2044 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text#3}}%
2045 }%
2046 }

```

4.10.2 Displaying entry details without adding information to the glossary

These commands merely display entry information without adding entries in the associated file or having hyperlinks.

Get the entry name (as specified by the `name` key when the entry was defined). The argument is the label associated with the entry. Note that unless you used `name=false` in the `sanitize` package option you may get unexpected results if the `name` key contains any commands.

```

\glsentryname
2047 \newcommand*{\glsentryname}[1]{\csname glo@#1@name\endcsname}

\Glsentryname
2048 \newcommand*{\Glsentryname}[1]{\%
2049 \protected@edef\@glo@text{\csname glo@#1@name\endcsname}%
2050 \expandafter\makefirstuc\expandafter{\@glo@text}}

```

Get the entry description (as specified by the `description` when the entry was defined). The argument is the label associated with the entry. Note that unless you used `description=false` in the `sanitize` package option you may get unexpected results if the `description` key contained any commands.

```

\glsentrydesc
2051 \newcommand*{\glsentrydesc}[1]{\csname glo@#1@desc\endcsname}

\Glsentrydesc
2052 \newcommand*{\Glsentrydesc}[1]{\%
2053 \protected@edef\@glo@text{\csname glo@#1@desc\endcsname}%
2054 \expandafter\makefirstuc\expandafter{\@glo@text}}

```

Plural form:

`\glentrydescplural`

```
2055 \newcommand*{\glentrydescplural}[1]{%
2056 \csname glo@#1@descplural\endcsname}
```

`\Glsentrydescplural`

```
2057 \newcommand*{\Glsentrydescplural}[1]{%
2058 \protected@edef\@glo@text{\csname glo@#1@descplural\endcsname}%
2059 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Get the entry text, as specified by the text key when the entry was defined.
The argument is the label associated with the entry:

`\glentrytext`

```
2060 \newcommand*{\glentrytext}[1]{\csname glo@#1@text\endcsname}
```

`\Glsentrytext`

```
2061 \newcommand*{\Glsentrytext}[1]{%
2062 \protected@edef\@glo@text{\csname glo@#1@text\endcsname}%
2063 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Get the plural form:

`\glentryplural`

```
2064 \newcommand*{\glentryplural}[1]{\csname glo@#1@plural\endcsname}
```

`\Glsentryplural`

```
2065 \newcommand*{\Glsentryplural}[1]{%
2066 \protected@edef\@glo@text{\csname glo@#1@plural\endcsname}%
2067 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Get the symbol associated with this entry. The argument is the label associated with the entry. Note that unless you used `symbol=false` in the `sanitize` package option you may get unexpected results if the `symbol` key contained any commands.

`\glentrysymbol`

```
2068 \newcommand*{\glentrysymbol}[1]{\csname glo@#1@symbol\endcsname}
```

`\Glsentrysymbol`

```
2069 \newcommand*{\Glsentrysymbol}[1]{%
2070 \protected@edef\@glo@text{\csname glo@#1@symbol\endcsname}%
2071 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Plural form:

`\glentrysymbolplural`

```
2072 \newcommand*{\glentrysymbolplural}[1]{%
2073 \csname glo@#1@symbolplural\endcsname}
```

`\Glsentrysymbolplural`

```
2074 \newcommand*{\Glsentrysymbolplural}[1]{%
2075 \protected@edef\@glo@text{\csname glo@#1@symbolplural\endcsname}%
2076 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Get the entry text to be used when the entry is first used in the document (as specified by the first key when the entry was defined).

`\glsentryfirst`

```
2077 \newcommand*{\glsentryfirst}[1]{\csname glo@#1@first\endcsname}
```

`\Glsentryfirst`

```
2078 \newcommand*{\Glsentryfirst}[1]{%
2079 \protected@edef\@glo@text{\csname glo@#1@first\endcsname}%
2080 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Get the plural form (as specified by the firstplural key when the entry was defined).

`\glsentryfirstplural`

```
2081 \newcommand*{\glsentryfirstplural}[1]{%
2082 \csname glo@#1@firstpl\endcsname}
```

`\Glsentryfirstplural`

```
2083 \newcommand*{\Glsentryfirstplural}[1]{%
2084 \protected@edef\@glo@text{\csname glo@#1@firstpl\endcsname}%
2085 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

Display the glossary type with which this entry is associated (as specified by the type key used when the entry was defined)

`\glsentrytype`

```
2086 \newcommand*{\glsentrytype}[1]{\csname glo@#1@type\endcsname}
```

Display the sort text used for this entry. Note that the `sort` key is sanitized, so unexpected results may occur if the `sort` key contained commands.

`\glsentrysort`

```
2087 \newcommand*{\glsentrysort}[1]{\csname glo@#1@sort\endcsname}
```

`\glsentryuseri` Get the first user key (as specified by the `user1` when the entry was defined). The argument is the label associated with the entry.

```
2088 \newcommand*{\glsentryuseri}[1]{\csname glo@#1@useri\endcsname}
```

`\Glsentryuseri`

```
2089 \newcommand*{\Glsentryuseri}[1]{%
2090 \protected@edef\@glo@text{\csname glo@#1@useri\endcsname}%
2091 \expandafter\makefirstuc\expandafter{\@glo@text}}
```

`\glentryuserii` Get the second user key (as specified by the `user2` when the entry was defined). The argument is the label associated with the entry.

```

2092 \newcommand*{\glentryuserii}[1]{\csname glo@#1@userii\endcsname}

\Glsentryuserii
2093 \newcommand*{\Glsentryuserii}[1]{%
2094 \protected@edef\@glo@text{\csname glo@#1@userii\endcsname}%
2095 \expandafter\makefirstuc\expandafter{\@glo@text}}

\glentryuseriii Get the third user key (as specified by the user3 when the entry was defined). The argument is the label associated with the entry.
2096 \newcommand*{\glentryuseriii}[1]{\csname glo@#1@useriii\endcsname}

\Glsentryuseriii
2097 \newcommand*{\Glsentryuseriii}[1]{%
2098 \protected@edef\@glo@text{\csname glo@#1@useriii\endcsname}%
2099 \expandafter\makefirstuc\expandafter{\@glo@text}}

\glentryuseriv Get the fourth user key (as specified by the user4 when the entry was defined). The argument is the label associated with the entry.
2100 \newcommand*{\glentryuseriv}[1]{\csname glo@#1@useriv\endcsname}

\Glsentryuseriv
2101 \newcommand*{\Glsentryuseriv}[1]{%
2102 \protected@edef\@glo@text{\csname glo@#1@useriv\endcsname}%
2103 \expandafter\makefirstuc\expandafter{\@glo@text}}

\glentryuserv Get the fifth user key (as specified by the user5 when the entry was defined). The argument is the label associated with the entry.
2104 \newcommand*{\glentryuserv}[1]{\csname glo@#1@userv\endcsname}

\Glsentryuserv
2105 \newcommand*{\Glsentryuserv}[1]{%
2106 \protected@edef\@glo@text{\csname glo@#1@userv\endcsname}%
2107 \expandafter\makefirstuc\expandafter{\@glo@text}}

\glentryuservi Get the sixth user key (as specified by the user6 when the entry was defined). The argument is the label associated with the entry.
2108 \newcommand*{\glentryuservi}[1]{\csname glo@#1@uservi\endcsname}

\Glsentryuservi
2109 \newcommand*{\Glsentryuservi}[1]{%
2110 \protected@edef\@glo@text{\csname glo@#1@uservi\endcsname}%
2111 \expandafter\makefirstuc\expandafter{\@glo@text}}

```

`\gls hyperlink` Provide a hyperlink to a glossary entry without adding information to the glossary file. The entry needs to be added using a command like `\gls link` or `\gls add` to ensure that the target is defined. The first (optional) argument specifies the link text. The entry name is used by default. The second argument is the entry label.

```
2112 \newcommand*{\gls hyperlink}[2][\glsentryname{\@glo@label}]{%
2113 \def\@glo@label{#2}%
2114 \@gls link{glo:#2}{#1}}
```

4.11 Adding an entry to the glossary without generating text

The following keys are provided for `\gls add` and `\gls addall`:

```
2115 \define@key{gloss add}{counter}{\def\@gls@counter{#1}}
2116 \define@key{gloss add}{format}{\def\@gls number format{#1}}
```

This key is only used by `\gls addall`:

```
2117 \define@key{gloss add}{types}{\def\@glo@type{#1}}
```

`\gls add[<options>]{<label>}`

Add a term to the glossary without generating any link text. The optional argument indicates which counter to use, and how to format it (using a key-value list) the second argument is the entry label. Note that *<options>* only has two keys: `counter` and `format` (the `types` key will be ignored).

`\gls add`

```
2118 \newcommand*{\gls add}[2][{}]{%
2119 \gls do if exists{#2}{%
2120 \def\@gls number format{gls number format}%
2121 \edef\@gls@counter{\csname glo@#2@counter\endcsname}%
2122 \setkeys{gloss add}{#1}%
2123 \edef\theglsentrycounter{\expandafter\noexpand
2124 \csname the\@gls@counter\endcsname}%
2125 \@do@wrglossary{#2}%
2126 }}
```

`\gls addall[<glossary list>]`

Add all terms defined for the listed glossaries (without displaying any text). If `types` key is omitted, apply to all glossary types.

`\gls addall`

```
2127 \newcommand*{\gls addall}[1][{}]{%
2128 \edef\@glo@type{\@glo@types}%
2129 \setkeys{gloss add}{#1}%
2130 \forallglsentries[\@glo@type]{\@glo@entry}{%
2131 \gls add{#1}{\@glo@entry}%
2132 }
```

4.12 Creating associated files

The `\writeist` command creates the associated customized `.ist` `makeindex` style file. While defining this command, some characters have their catcodes temporarily changed to ensure they get written to the `.ist` file correctly. The `makeindex` actual character (usually `@`) is redefined to be a `?`, to allow internal commands to be written to the glossary file output file.

The special characters are stored in `\@gls@actualchar`, `\@gls@encapchar`, `\@gls@levelchar` and `\@gls@quotechar` to make them easier to use later, but don't change these values, because the characters are encoded in the command definitions that are used to escape the special characters (which means that the user no longer needs to worry about `makeindex` special characters).

The symbols and numbers label for group headings are hardwired into the `.ist` file as `glsymbols` and `glsnumbers`, the group titles can be translated (so that `\glsymbolsgroupname` replaces `glsymbols` and `\glsnumbersgroupname` replaces `glsnumbers`) using the command `\glsgetgrouptitle` which is defined in `.` This is done to prevent any problem characters in `\glsymbolsgroupname` and `\glsnumbersgroupname` from breaking hyperlinks.

`\glsopenbrace` Define `\glsopenbrace` to make it easier to write an opening brace to a file.

```
2133 \edef\glsopenbrace{\expandafter\@gobble\string\{}
```

`\glsclosebrace` Define `\glsclosebrace` to make it easier to write an opening brace to a file.

```
2134 \edef\glsclosebrace{\expandafter\@gobble\string\}}
```

`\glsquote` Define command that makes it easier to write quote marks to a file in the event that the double quote character has been made active.

```
2135 \edef\glsquote#1{\string"#1\string"}
```

`\@glsfirstletter` Define the first letter to come after the digits 0,...,9. Only required for `xindy`.

```
2136 \ifglsxindy
```

```
2137 \newcommand*{\@glsfirstletter}{A}
```

```
2138 \fi
```

`GlsSetXdyFirstLetterAfterDigits` Sets the first letter to come after the digits 0,...,9.

```
2139 \ifglsxindy
```

```
2140 \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
```

```
2141 \renewcommand*{\@glsfirstletter}{#1}}
```

```
2142 \else
```

```
2143 \newcommand*{\GlsSetXdyFirstLetterAfterDigits}[1]{%
```

```
2144 \glsnoxywarning\GlsSetXdyFirstLetterAfterDigits}
```

```
2145 \fi
```

`\@glsminrange` Define the minimum number of successive location references to merge into a range.

```
2146 \newcommand*{\@glsminrange}{2}
```

`\GlsSetXdyMinRangeLength` Set the minimum range length. The value must either be `none` or a positive integer. The glossaries package doesn't check if the argument is valid, that is left to `xindy`.

```

2147 \ifglsxindy
2148   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
2149     \renewcommand*{\@glxminrange}{#1}}
2150 \else
2151   \newcommand*{\GlsSetXdyMinRangeLength}[1]{%
2152     \glsnxindywarning\GlsSetXdyMinRangeLength}
2153 \fi

```

`\writeist`

```

2154 \newwrite\istfile
2155 \ifglsxindy
  Code to use if xindy is required.
2156   \def\writeist{%
    Open the style file
2157     \openout\istfile=\istfilename
    Write header comment at the start of the file
2158     \write\istfile{;; xindy style file created by the glossaries
2159       package}%
2160     \write\istfile{;; for document '\jobname' on
2161       \the\year-\the\month-\the\day}%
    Specify the required styles
2162     \write\istfile{^^J; required styles^^J}
2163     \@for\@xdystyle:=\@xdyrequiredstyles\do{%
2164       \ifx\@xdystyle\@empty
2165       \else
2166         \protected@write\istfile{}{(require
2167           \string"\@xdystyle.xdy\string")}%
2168       \fi
2169     }%
    List the allowed attributes (possible values used by the format key)
2170     \write\istfile{^^J%
2171       ; list of allowed attributes (number formats)^^J}%
2172     \write\istfile{(define-attributes ((\@xdyattributes)))}%
    Define any additional alphabets
2173     \write\istfile{^^J; user defined alphabets^^J}%
2174     \write\istfile{\@xdyuseralphabets}%
    Define location classes.
2175     \write\istfile{^^J; location class definitions^^J}%
    Lower case Roman numerals (i, ii, ...). In the event that \roman has been re-
    defined to produce a fancy form of roman numerals, attempt to work out how it will
    be written to the output file.
2176     \protected@edef\@gls@roman{\@roman{0}\string"

```

```

2177     \string"roman-numbers-lowercase\string" :sep \string"}}%
2178     \@onelevel@sanitize\@gls@roman
2179     \edef\@tmp{\string" \string"roman-numbers-lowercase\string"
2180       :sep \string"}}%
2181     \@onelevel@sanitize\@tmp
2182     \ifx\@tmp\@gls@roman
2183       \write\istfile{(define-location-class
2184         \string"roman-page-numbers\string"^^J\space\space\space
2185         (\string"roman-numbers-lowercase\string")
2186         :min-range-length \@glsminrange)}}%
2187     \else
2188       \write\istfile{(define-location-class
2189         \string"roman-page-numbers\string"^^J\space\space\space
2190         (:sep "\@gls@roman")
2191         :min-range-length \@glsminrange)}}%
2192     \fi

```

Upper case Roman numerals (I, II, ...)

```

2193     \write\istfile{(define-location-class
2194       \string"Roman-page-numbers\string"^^J\space\space\space
2195       (\string"roman-numbers-uppercase\string")
2196       :min-range-length \@glsminrange)}}%

```

Arabic numbers (1, 2, ...)

```

2197     \write\istfile{(define-location-class
2198       \string"arabic-page-numbers\string"^^J\space\space\space
2199       (\string"arabic-numbers\string")
2200       :min-range-length \@glsminrange)}}%

```

Lower case alphabetical locations (a, b, ...)

```

2201     \write\istfile{(define-location-class
2202       \string"alpha-page-numbers\string"^^J\space\space\space
2203       (\string"alpha\string")
2204       :min-range-length \@glsminrange)}}%

```

Upper case alphabetical locations (A, B, ...)

```

2205     \write\istfile{(define-location-class
2206       \string"Alpha-page-numbers\string"^^J\space\space\space
2207       (\string"ALPHA\string")
2208       :min-range-length \@glsminrange)}}%

```

Appendix style locations (e.g. A-1, A-2, ..., B-1, B-2, ...). The separator is given by \@glsAlphacompositor.

```

2209     \write\istfile{(define-location-class
2210       \string"Appendix-page-numbers\string"^^J\space\space\space
2211       (\string"ALPHA\string"
2212       :sep \string"\@glsAlphacompositor\string"
2213       \string"arabic-numbers\string")
2214       :min-range-length \@glsminrange)}}%

```

Section number style locations (e.g. 1.1, 1.2, ...). The compositor is given by \glscompositor.


```

2215 \write\istfile{(define-location-class
2216 \string"arabic-section-numbers\string"^^J\space\space\space
2217 (\string"arabic-numbers\string"
2218 :sep \string"\glscompositor\string"
2219 \string"arabic-numbers\string")
2220 :min-range-length \@glsminrange)}}%

```

User defined location classes.

```

2221 \write\istfile{^^J; user defined location classes}%
2222 \write\istfile{\@xdyuserlocationdefs}%

```

Cross-reference class. (The unverified option is used as the cross-references are supplied using the list of labels along with the optional argument for `\glsseeformat` which xindy won't recognise.)

```

2223 \write\istfile{^^J; define cross-reference class^^J}%
2224 \write\istfile{(define-crossref-class \string"see\string"
2225 :unverified )}%

```

Define how cross-references should be displayed. This adds an empty set of braces after the cross-referencing information allowing for the final argument of `\glsseeformat` which gets ignored. (When using `makeindex` this final argument contains the location information which is not required.)

```

2226 \write\istfile{(markup-crossref-list
2227 :class \string"see\string"^^J\space\space\space
2228 :open \string"\string\glsseeformat\string"
2229 :close \string"{}\string")}%

```

List the order to sort the classes.

```

2230 \write\istfile{^^J; define the order of the location classes}%
2231 \write\istfile{(define-location-class-order
2232 (\@xdylocationclassorder))}%

```

Specify what to write to the start and end of the glossary file.

```

2233 \write\istfile{^^J; define the glossary markup^^J}%
2234 \write\istfile{(markup-index^^J\space\space\space
2235 :open \string"\string
2236 \glossarysection[\string\glossarytoctitle]{\string
2237 \glossarytitle}\string\glossarypreamble\string~n\string\begin
2238 {theglossary}\string\glossaryheader\string~n\string" ^^J\space
2239 \space\space:close \string"\expandafter\@gobble
2240 \string%\string~n\string
2241 \end{theglossary}\string\glossarypostamble
2242 \string~n\string" ^^J\space\space\space
2243 :tree)}}%

```

Specify what to put between letter groups

```

2244 \write\istfile{(markup-letter-group-list
2245 :sep \string"\string\glsgroupskip\string~n\string")}%

```

Specify what to put between entries

```

2246 \write\istfile{(markup-indexentry
2247 :open \string"\string\relax \string\glsresetentrylist
2248 \string~n\string")}%

```

Specify how to format entries

```
2249 \write\istfile{(markup-locclass-list :open
2250 \string\glsopenbrace\string\glossaryentrynumbers
2251 \glsopenbrace\string\relax\space \string^^J\space\space\space
2252 :sep \string", \string"
2253 :close \string"\glsclosebrace\glsclosebrace\string"))%
```

Specify how to separate location numbers

```
2254 \write\istfile{(markup-locref-list
2255 :sep \string\string\delimN\space\string"))%
```

Specify how to indicate location ranges

```
2256 \write\istfile{(markup-range
2257 :sep \string\string\delimR\space\string"))%
```

Specify 2-page and 3-page suffixes, if defined. First, the values must be sanitized to write them explicitly.

```
2258 \@onelevel@sanitize\gls@suffiF
2259 \@onelevel@sanitize\gls@suffiFF
2260 \ifx\gls@suffiF\@empty
2261 \else
2262 \write\istfile{(markup-range
2263 :close "\gls@suffiF" :length 1 :ignore-end))%
2264 \fi
2265 \ifx\gls@suffiFF\@empty
2266 \else
2267 \write\istfile{(markup-range
2268 :close "\gls@suffiFF" :length 2 :ignore-end))%
2269 \fi
```

Specify how to format locations.

```
2270 \write\istfile{^^J; define format to use for locations^^J}%
2271 \write\istfile{\@xdylocref}%
```

Specify how to separate letter groups.

```
2272 \write\istfile{^^J; define letter group list format^^J}%
2273 \write\istfile{(markup-letter-group-list
2274 :sep \string\string\glsgroupskip\string~n\string"))%
```

Define letter group headings.

```
2275 \write\istfile{^^J; letter group headings^^J}%
2276 \write\istfile{(markup-letter-group
2277 :open-head \string\string\glsgroupheading
2278 \glsopenbrace\string^^J\space\space\space
2279 :close-head \string\glsclosebrace\string"))%
```

Define additional letter groups.

```
2280 \write\istfile{^^J; additional letter groups^^J}%
2281 \write\istfile{\@xdylettergroups}%
```

Define additional sort rules

```
2282 \write\istfile{^^J; additional sort rules^^J}
2283 \write\istfile{\@xdysortrules}%
2284 \noist}
2285 \else
```

Code to use if makeindex is required.

```
2286 \edef\@gls@actualchar{\string?}
2287 \edef\@gls@encapchar{\string|}
2288 \edef\@gls@levelchar{\string!}
2289 \edef\@gls@quotechar{\string"}
2290 \def\writeist{\relax
2291   \openout\istfile=\istfilename
2292   \write\istfile{\expandafter\@gobble\string\% makeindex style file
2293     created by the glossaries package}
2294   \write\istfile{\expandafter\@gobble\string\% for document
2295     'jobname' on \the\year-\the\month-\the\day}
2296   \write\istfile{actual '\@gls@actualchar'}
2297   \write\istfile{encap '\@gls@encapchar'}
2298   \write\istfile{level '\@gls@levelchar'}
2299   \write\istfile{quote '\@gls@quotechar'}
2300   \write\istfile{keyword \string"\string\glossaryentry\string"}
2301   \write\istfile{preamble \string"\string\glossarysection[\string
2302     \glossarytoctitle]{\string\glossarytitle}\string
2303     \glossarypreamble\string\n\string\begin{theglossary}\string
2304     \glossaryheader\string\n\string"}
2305   \write\istfile{postamble \string"\string%\string\n\string
2306     \end{theglossary}\string\glossarypostamble\string\n
2307     \string"}
2308   \write\istfile{group_skip \string"\string\glsgroupskip\string\n
2309     \string"}
2310   \write\istfile{item_0 \string"\string%\string\n\string"}
2311   \write\istfile{item_1 \string"\string%\string\n\string"}
2312   \write\istfile{item_2 \string"\string%\string\n\string"}
2313   \write\istfile{item_01 \string"\string%\string\n\string"}
2314   \write\istfile{item_x1
2315     \string"\string\relax \string\glsresetentrylist\string\n
2316     \string"}
2317   \write\istfile{item_12 \string"\string%\string\n\string"}
2318   \write\istfile{item_x2
2319     \string"\string\relax \string\glsresetentrylist\string\n
2320     \string"}

2321   \write\istfile{delim_0 \string"\string\{\string
2322     \glossaryentrynumbers\string\{\string\relax \string"}
2323   \write\istfile{delim_1 \string"\string\{\string
2324     \glossaryentrynumbers\string\{\string\relax \string"}
2325   \write\istfile{delim_2 \string"\string\{\string
2326     \glossaryentrynumbers\string\{\string\relax \string"}
2327   \write\istfile{delim_t \string"\string\}\string\}\string"}
```

```

2328 \write\istfile{delim_n \string\string\delimN \string}
2329 \write\istfile{delim_r \string\string\delimR \string}
2330 \write\istfile{headings_flag 1}
2331 \write\istfile{heading_prefix
2332 \string\string\glsgroupheading\string\{\string}
2333 \write\istfile{heading_suffix
2334 \string\string\}\string\relax
2335 \string\glsgroupresetentrylist \string}
2336 \write\istfile{symhead_positive \string"glssymbols\string"}
2337 \write\istfile{numhead_positive \string"glslnumbers\string"}
2338 \write\istfile{page_compositor \string"glscpositor\string"}
2339 \@gls@escbsdq\gls@suffixF
2340 \@gls@escbsdq\gls@suffixFF
2341 \ifx\gls@suffixF\@empty
2342 \else
2343 \write\istfile{suffix_2p \string"\gls@suffixF\string"}
2344 \fi
2345 \ifx\gls@suffixFF\@empty
2346 \else
2347 \write\istfile{suffix_3p \string"\gls@suffixFF\string"}
2348 \fi
2349 \noist
2350 }
2351 \fi

```

The command `\noist` will suppress the creation of the `.ist` file (it simply redefines `\writeist` to do nothing). Obviously you need to use this command before `\writeist` to have any effect. Since the `.ist` file should only be created once, `\noist` is called at the end of `\writeist`.

`\noist`

```

2352 \newcommand{\noist}{\let\writeist\relax}

```

`\@makeglossary` is an internal command that takes an argument indicating the glossary type. This command will create the glossary file required by `makeindex` for the given glossary type, using the extension supplied by the `<out-ext>` parameter used in `\newglossary` (and it will also activate the `\glossary` command, and create the customized `.ist` `makeindex` style file).

Note that you can't use `\@makeglossary` for only some of the defined glossaries. You either need to have a `\makeglossary` for all glossaries or none (otherwise you will end up with a situation where \TeX is trying to write to a non-existent file). The relevant glossary must be defined prior to using `\@makeglossary`.

`\@makeglossary`

```

2353 \newcommand*{\@makeglossary}[1]{%
2354 \ifglossaryexists{#1}{%
2355 \edef\glo@out{\csname @glotype@#1\out\endcsname}%
2356 \expandafter\newwrite\csname glo@#1\file\endcsname
2357 \edef\glo@file{\csname glo@#1\file\endcsname}%

```

```

2358 \immediate\openout\@glo@file=\jobname.\glo@out
2359 \@gls@renewglossary
2360 \PackageInfo{glossaries}{Writing glossary file \jobname.\glo@out}
2361 \writeist
2362 }\PackageError{glossaries}{%
2363 Glossary type ‘#1’ not defined}{New glossaries must be defined before
2364 using \string\makeglossary}}

```

`\warn@nomakeglossaries` Issue warning that `\makeglossaries` hasn't been used.

```

2365 \newcommand*{\warn@nomakeglossaries}{%
2366   \GlossariesWarningNoLine{\string\makeglossaries\space
2367   hasn't been used,^~Jthe glossaries will not be updated}%
2368 }

```

`\makeglossaries` will use `\@makeglossary` for each glossary type that has been defined. New glossaries need to be defined before using `\makeglossary`, so have `\makeglossaries` redefine `\newglossary` to prevent it being used afterwards.

`\makeglossaries`

```

2369 \newcommand*{\makeglossaries}{%
2370 % Write the name of the style file to the aux file
2371 % (needed by \app{makeglossaries})
2372 %   \begin{macrocode}
2373   \protected@write\@auxout{}{\string\@istfilename{\istfilename}}%
2374   \protected@write\@auxout{}{\string\@glsorder{\glsorder}}

```

Iterate through each glossary type and activate it.

```

2375   \@for\@glo@type:=\@glo@types\do{%
2376     \ifthenelse{\equal{\@glo@type}{}}{}{}%
2377     \@makeglossary{\@glo@type}%
2378   }%

```

New glossaries must be created before `\makeglossaries` so disable `\newglossary`.

```

2379   \renewcommand*\newglossary[4][]{%
2380   \PackageError{glossaries}{New glossaries
2381   must be created before \string\makeglossaries}{You need
2382   to move \string\makeglossaries\space after all your
2383   \string\newglossary\space commands}}%

```

Any subsequence instances of this command should have no effect

```

2384   \let\@makeglossary\relax
2385   \let\makeglossary\relax
2386   \let\makeglossaries\relax

```

Disable all commands that have no effect after `\makeglossaries`

```

2387   \@disable@onlypremakeg
2388   Suppress warning about no \makeglossaries
2388   \let\warn@nomakeglossaries\relax
2389 }

```

The `\makeglossary` command is redefined to be identical to `\makeglossaries`. (This is done to reinforce the message that you must either use `\@makeglossary` for all the glossaries or for none of them.)

`\makeglossary`

```
2390 \let\makeglossary\makeglossaries
```

If `\makeglossaries` hasn't been used, issue a warning. Also issue a warning if neither `\printglossaries` nor `\printglossary` have been used.

```
2391 \AtEndDocument{%
2392   \warn@nomakeglossaries
2393   \warn@noprintglossary
2394 }
```

4.13 Writing information to associated files

The `\glossary` command is redefined so that it takes an optional argument *<type>* to specify the glossary type (use `\glsdefaulttype` glossary by default). This shouldn't be used at user level as `\glslink` sets the correct format. The associated number should be stored in `\theglsentrycounter` before using `\glossary`.

`\glossary`

```
2395 \renewcommand*{\glossary}[1][\glsdefaulttype]{%
2396 \glossary[#1]}
```

Define internal `\@glossary` to ignore its argument. This gets redefined in `\@makeglossary`. This is defined to just `\index` as memoir changes the definition of `\@index`. (Thanks to Dan Luecking for pointing this out.)

`\@glossary`

```
2397 \def\@glossary[#1]{\index}
```

This is a convenience command to set `\@glossary`. It is used by `\@makeglossary` and then redefined to do nothing, as it only needs to be done once.

`\@gls@renewglossary`

```
2398 \newcommand{\@gls@renewglossary}{%
2399 \gdef\@glossary[##1]{\@bsphack\begingroup\@wrglossary{##1}}%
2400 \let\@gls@renewglossary\@empty
2401 }
```

The `\@wrglossary` command is redefined to have two arguments. The first argument is the glossary type, the second argument is the glossary entry (the format of which is set in `\glslink`).

`\@wrglossary`

```
2402 \renewcommand*{\@wrglossary}[2]{%
2403   \expandafter\protected@write\csname glo@#1@file\endcsname\{#\2}%
2404   \endgroup\@esphack
2405 }
```

`\do@wrglossary` Write the glossary entry in the appropriate format. (Need to set `glsnumberformat` and `gls@counter` prior to use.) The argument is the entry’s label.

2406 `\newcommand{\do@wrglossary}[1]{%`

Determine whether to use `xindy` or `makeindex` syntax

2407 `\ifglxindy`

Need to determine if the formatting information starts with a (or) indicating a range.

2408 `\expandafter\@glo@check@mkidxrangechar\@glsnumberformat\@nil`

2409 `\def\@glo@range{}%`

2410 `\expandafter\if\@glo@prefix(\relax`

2411 `\def\@glo@range{:open-range}%`

2412 `\else`

2413 `\expandafter\if\@glo@prefix)\relax`

2414 `\def\@glo@range{:close-range}%`

2415 `\fi`

2416 `\fi`

Get the location and escape any special characters

2417 `\protected@edef\@glslocref{\theglsentrycounter}%`

2418 `\@gls@checkmkidxchars\@glslocref`

Write to the glossary file using `xindy` syntax.

2419 `\glossary[\csname glo@#1@type\endcsname]{%`

2420 `(indexentry :tkey (\csname glo@#1@index\endcsname)`

2421 `:locref \string"\@glslocref\string" %`

2422 `:attr \string"\@glo@suffix\string" \@glo@range`

2423 `)`

2424 `}%`

2425 `\else`

Convert the format information into the format required for `makeindex`

2426 `\@set@glo@numformat\@glo@numfmt\@gls@counter\@glsnumberformat`

Write to the glossary file using `makeindex` syntax.

2427 `\glossary[\csname glo@#1@type\endcsname]{%`

2428 `\string\glossaryentry{\csname glo@#1@index\endcsname`

2429 `\@gls@encapchar\@glo@numfmt}{\theglsentrycounter}}%`

2430 `\fi`

2431 `}`

4.14 Glossary Entry Cross-References

`\do@seeglossary` Write the glossary entry with a cross reference. The first argument is the entry’s label, the second must be in the form `[\langle tag \rangle]{\langle list \rangle}`, where `\langle tag \rangle` is a tag such as “see” and `\langle list \rangle` is a list of labels.

2432 `\newcommand{\do@seeglossary}[2]{%`

2433 `\ifglxindy`

2434 `\glossary[\csname glo@#1@type\endcsname]{%`

2435 `(indexentry`

```

2436      :tkey (\csname glo@#1@index\endcsname)
2437      :xref (\string"#2\string")
2438      :attr \string"see\string"
2439    )
2440  }%
2441 \else
2442   \glossary[\csname glo@#1@type\endcsname]{%
2443   \string\glossaryentry{\csname glo@#1@index\endcsname
2444   \@gls@encapchar glsseeformat#2}{Z}}%
2445 \fi
2446 }

```

`\@gls@fixbraces` If no optional argument is specified, list needs to be enclosed in a set of braces.

```

2447 \def\@gls@fixbraces#1#2#3\@nil{%
2448   \ifx#2[\relax
2449     \def#1{#2#3}%
2450   \else
2451     \def#1{{#2#3}}%
2452   \fi
2453 }

```

`\glssee` `\glssee{<label>}{<cross-ref list>}`

```

2454 \newcommand*{\glssee}[3][\seename]{%
2455   \@do@seeglossary{#2}{[#1]{#3}}
2456 \newcommand*{\@glssee}[3][\seename]{%
2457   \glssee[#1]{#3}{#2}}
2458 %   \end{macrocode}
2459 %\end{macro}
2460 %
2461 %\begin{macro}{\glsseeformat}
2462 %\changes{1.17}{2008 December 26}{new}
2463 % The first argument specifies what tag to use (e.g.\ ‘‘see’’),
2464 % the second argument is a comma-separated list of labels.
2465 % The final argument (the location) is ignored.
2466 %   \begin{macrocode}
2467 \newcommand*{\glsseeformat}[3][\seename]{\emph{#1} \glsseelist{#2}}

```

`\glsseelist` `\glsseelist{<list>}` formats list of entry labels.

```

2468 \newcommand*{\glsseelist}[1]{%

```

If there is only one item in the list, set the last separator to do nothing.

```

2469   \let\@gls@dolast\relax

```

Don't display separator on the first iteration of the loop

```

2470   \let\@gls@donext\relax

```

Iterate through the labels

```

2471   \@for\@gls@thislabel:=#1\do{%

```

Check if on last iteration of loop

```

2472     \ifx\@xfor@nextelement\@nnil

```



```

2473     \@gls@dolast
2474     \else
2475     \@gls@donext
2476     \fi
    display the entry for this label
2477     \glsseeitem{\@gls@thislabel}%
    Update separators
2478     \let\@gls@dolast\glsseelastsep
2479     \let\@gls@donext\glsseesep
2480 }%
2481 }

```

`\glsseelastsep` Separator to use between penultimate and ultimate entries in a cross-referencing list.

```
2482 \newcommand*{\glsseelastsep}{\space\andname\space}
```

`\glsseesep` Separator to use between entries in a cross-referencing list.

```
2483 \newcommand*{\glsseesep}{, }
```

`\glsseeitem` `\glsseeitem{<label>}` formats individual entry in a cross-referencing list.

```
2484 \newcommand*{\glsseeitem}[1]{\glshyperlink{#1}}
```

4.15 Displaying the glossary

An individual glossary is displayed in the text using `\printglossary[<key-val list>]`. If the `type` key is omitted, the default glossary is displayed. The optional argument can be used to specify an alternative glossary, and can also be used to set the style, title and entry in the table of contents. Available keys are defined below.

`\warn@noprntglossary` Warn the user if they have forgotten `\printglossaries` or `\printglossary`. (Will be suppressed if there is at least one occurrence of `\printglossary`. There is no check to ensure that there is a `\printglossary` for each defined glossary.)

```

2485 \def\warn@noprntglossary{\GlossariesWarningNoLine{No
2486   \string\printglossary\space or \string\printglossaries\space
2487   found.^^JThis document will not have a glossary}}

```

`\printglossary` The TOC title needs to be processed in a different manner to the main title in case the `translator` and `hyperref` packages are both being used.

```
2488 \@ifundefined{printglossary}{}{%
```

If `\printglossary` is already defined, issue a warning and undefine it.

```

2489   \GlossariesWarning{Overriding \string\printglossary}%
2490   \let\printglossary\undefined
2491 }

```

`\printglossary` has an optional argument. The default value is to set the glossary type to the main glossary.

```
2492 \newcommand*{\printglossary}[1][type=\glsdefaulttype]{%
```

If `xindy` is being used, need to find the root language for `makeglossaries` to pass to `xindy`.

```
2493 \ifglxindy\findrootlanguage\fi
```

Set up defaults.

```
2494 \def\@glo@type{\glsdefaulttype}%
2495 \def\glossarytitle{\csname @glotype@\@glo@type @title\endcsname}%
2496 \def\@glossarystyle{}%
2497 \def\gls@dotoc@title{\glssettoctitle{\@glo@type}}%
```

Store current value of `\glossaryentrynumbers`. (This may be changed via the optional argument)

```
2498 \let\@org@glossaryentrynumbers\glossaryentrynumbers
```

Localise the effects of the optional argument

```
2499 \bgroup
```

Determine settings specified in the optional argument.

```
2500 \setkeys{printgloss}{#1}%
```

Enable individual number lists to be suppressed.

```
2501 \let\org@glossaryentrynumbers\glossaryentrynumbers
2502 \let\glsnonextpages\glsnonextpages
```

Enable suppression of description terminators.

```
2503 \let\nopostdesc\@nopostdesc
```

Set up the entry for the TOC

```
2504 \gls@dotoc@title
```

Set the glossary style

```
2505 \@glossarystyle
```

Some macros may end up being expanded into internals in the glossary, so need to make `@` a letter.

```
2506 \makeatletter
```

Input the glossary file, if it exists.

```
2507 \@input@{\jobname.\csname @glotype@\@glo@type @in\endcsname}%
```

If the glossary file doesn't exist, do `\null`. (This ensures that the page is shipped out and all write commands are done.) This might produce an empty page, but at this point the document isn't complete, so it shouldn't matter.

```
2508 \IfFileExists{\jobname.\csname @glotype@\@glo@type @in\endcsname}{}%
2509 {\null}%
```

If `xindy` is being used, need to write the language dependent information to the `.aux` file for `makeglossaries`.

```
2510 \ifglxindy
2511 \ifundefined{xdy@\@glo@type @language}{%
```

```

2512     \protected@write\@auxout{}\{%
2513     \string\@xdylanguage{\@glo@type}\@xdy@main@language}}%
2514   }\{%
2515     \protected@write\@auxout{}\{%
2516     \string\@xdylanguage{\@glo@type}\@csname @xdy@\@glo@type
2517     @language\endcsname}}%
2518   }\%
2519     \protected@write\@auxout{}\{%
2520     \string\@gls@codepage{\@glo@type}\@gls@codepage}}%
2521   \fi
2522 \egroup
Reset \glossaryentrynumbers
2523 \global\let\glossaryentrynumbers\@org@glossaryentrynumbers
Suppress warning about no \printglossary
2524 \global\let\warn@noprintglossary\relax
2525 }

```

The `\printglossaries` command will do `\printglossary` for each glossary type that has been defined. It is better to use `\printglossaries` rather than individual `\printglossary` commands to ensure that you don't forget any new glossaries you may have created. It also makes it easier to chop and change the value of the `acronym` package option. However, if you want to list the glossaries in a different order, or if you want to set the title or table of contents entry, or if you want to use different glossary styles for each glossary, you will need to use `\printglossary` explicitly for each glossary type.

`\printglossaries`

```

2526 \newcommand*\printglossaries\{%
2527 \foralllglossaries{\@glo@type}\printglossary[type=\@glo@type]}

```

The keys that can be used in the optional argument to `\printglossary` are as follows: The `type` key sets the glossary type.

```

2528 \define@key{printgloss}{type}\def\@glo@type{#1}

```

The `title` key sets the title used in the glossary section header. This overrides the title used in `\newglossary`.

```

2529 \define@key{printgloss}{title}\def\glossarytitle{#1}

```

The `toctitle` sets the text used for the relevant entry in the table of contents.

```

2530 \define@key{printgloss}{toctitle}\def\glossarytoctitle{#1}%
2531 \let\gls@dotoc\relax
2532 }

```

The `style` key sets the glossary style (but only for the given glossary).

```

2533 \define@key{printgloss}{style}\{%
2534 \ifundefined{glsstyle@#1}\PackageError{glossaries}{Glossary
2535 style '#1' undefined}\{}{}\%
2536 \def\@glossarystyle{\csname @glsstyle@#1\endcsname}}

```

The `numberedsection` key determines if this glossary should be in a numbered section.

```

2537 \define@choicekey{printgloss}{numberedsection}[\val\nr]{%
2538 false,nolabel,autolabel}[nolabel]{%
2539 \ifcase\nr\relax
2540   \renewcommand*{\@@glossarysecstar}{*}%
2541   \renewcommand*{\@@glossaryseclabel}{}%
2542 \or
2543   \renewcommand*{\@@glossarysecstar}{}%
2544   \renewcommand*{\@@glossaryseclabel}{}%
2545 \or
2546   \renewcommand*{\@@glossarysecstar}{}%
2547   \renewcommand*{\@@glossaryseclabel}{\label{\glsautoprefix\@glo@type}}%
2548 \fi}

```

The `nonumberlist` key determines if this glossary should have a number list.

```

2549 \define@boolkey{printgloss}[gls]{nonumberlist}[true]{%
2550 \ifglslnonumberlist
2551   \def\glossaryentrynumbers##1{%
2552 \else
2553   \def\glossaryentrynumbers##1{##1}%
2554 \fi}

```

`\@glslnonextpages` Suppresses the next number list only. Global assignments required as it may not occur in the same level of grouping as the next numberlist. (For example, if `\glslnonextpages` is placed in the entry's description and 3 column tabular style glossary is used.) `\org@glossaryentrynumbers` needs to be set at the start of each glossary, in the event that `\glossaryentrynumber` is redefined.

```

2555 \newcommand*{\@glslnonextpages}{%
2556   \gdef\glossaryentrynumbers##1{%
2557     \glsresetentrylist}}

```

`\glsresetentrylist` Resets `\glossaryentrynumbers`

```

2558 \newcommand*{\glsresetentrylist}{%
2559   \global\let\glossaryentrynumbers\org@glossaryentrynumbers}

```

`\glslnonextpages` Outside of `\printglossary` this does nothing.

```

2560 \newcommand*{\glslnonextpages}{}

```

`theglossary` If the `theglossary` environment has already been defined, a warning will be issued. This environment should be redefined by glossary styles.

```

2561 \@ifundefined{theglossary}{%
2562   \newenvironment{theglossary}{}{}%
2563 }{%
2564   \GlossariesWarning{overriding 'theglossary' environment}%
2565   \renewenvironment{theglossary}{}{}%
2566 }

```

The glossary header is given by `\glossaryheader`. This forms part of the glossary style, and must indicate what should appear immediately after the start of the `\theglossary` environment. (For example, if the glossary uses a tabular-like environment, it may be used to set the header row.) Note that if you don't want a header row, the glossary style must redefine `\glossaryheader` to do nothing.

`\glossaryheader`

```
2567 \newcommand*\glossaryheader{}
```

`\glstarget` `\glstarget{<label>}{<name>}`

Provide user interface to `\@glstarget` to make it easier to modify the glossary style in the document.

```
2568 \newcommand*\glstarget[2]{\@glstarget{glo:#1}{#2}}
```

`\glossaryentryfield` `\glossaryentryfield{<label>}{<name>}{<description>}{<symbol>}{<page-list>}`

This command governs how each entry row should be formatted in the glossary. Glossary styles need to redefine this command. Most of the predefined styles ignore `<symbol>`.

```
2569 \newcommand*\glossaryentryfield[5]{%
```

```
2570 \noindent\textbf{\glstarget{#1}{#2}} #4 #3. #5\par}
```

`\glossaryentryfield` `\glossarysubentryfield{<level>}{<label>}{<name>}{<description>}{<symbol>}{<page-list>}`

This command governs how each subentry should be formatted in the glossary. Glossary styles need to redefine this command. Most of the predefined styles ignore `<symbol>`. The first argument is a number indicating the level. (The level should be greater than or equal to 1.)

```
2571 \newcommand*\glossarysubentryfield[6]{%
```

```
2572 \glstarget{#2}{\strut}#4. #6\par}
```

Within each glossary, the entries form distinct groups which are determined by the first character of the `sort` key. When using `makeindex`, there will be a maximum of 28 groups: symbols, numbers, and the 26 alphabetical groups A, ..., Z. If you use `xindy` the groups will depend on whatever alphabet is used. This is determined by the language or custom alphabets can be created in the `xindy` style file. The command `\glsgroupskip` specifies what to do between glossary groups. Glossary styles must redefine this command. (Note that `\glsgroupskip` only occurs between groups, not at the start or end of the glossary.)

`\glsgroupskip`

```
2573 \newcommand*\glsgroupskip{}
```

Each of the 28 glossary groups described above is preceded by a group heading. This is formatted by the command `\glsgroupheading` which takes one argument which is the *label* assigned to that group (not the title). The corresponding labels are: `glsymbols`, `glsnumbers`, A, ..., Z. Glossary styles must be redefined

this command. (In between groups, `\glsgroupheading` comes immediately after `\glsgroupskip`.)

`\glsgroupheading`

```
2574 \newcommand*{\glsgroupheading}[1]{}

```

It is possible to “trick” `makeindex` into treating entries as though they belong to the same group, even if the terms don’t start with the same letter, by modifying the `sort` key. For example, all entries belonging to one group could be defined so that the `sort` key starts with an `a`, while entries belonging to another group could be defined so that the `sort` key starts with a `b`, and so on. If you want each group to have a heading, you would then need to modify the translation control sequences `\glsgetgrouptitle` and `\glsgetgrouplabel` so that the label is translated into the required title (and vice-versa).

`\glsgetgrouptitle{<label>}`

This command produces the title for the glossary group whose label is given by `<label>`. By default, the group labelled `glsymbols` produces `\glssymbolsgroupname`, the group labelled `glsnumbers` produces `\glsnumbersgroupname` and all the other groups simply produce their label. As mentioned above, the group labels are: `glsymbols`, `glsnumbers`, `A`, `...`, `Z`. If you want to redefine the group titles, you will need to redefine this command.

`\glsgetgrouptitle`

```
2575 \newcommand*{\glsgetgrouptitle}[1]{%
2576 \@ifundefined{#1groupname}{#1}{\csname #1groupname\endcsname}}

```

`\glsgetgrouplabel{<title>}`

This command does the reverse to the previous command. The argument is the group title, and it produces the group label. Note that if you redefine `\glsgetgrouptitle`, you will also need to redefine `\glsgetgrouplabel`.

`\glsgetgrouplabel`

```
2577 \newcommand*{\glsgetgrouplabel}[1]{%
2578 \ifthenelse{\equals{#1}{\glssymbolsgroupname}}{\glssymbols}{%
2579 \ifthenelse{\equals{#1}{\glsnumbersgroupname}}{\glsnumbers}{#1}}

```

The command `\setentrycounter` sets the entry’s associated counter (required by `\glshypernumber` etc.) `\glslink` and `\glsadd` encode the `\glossary` argument so that the relevant counter is set prior to the formatting command.

`\setentrycounter`

```
2580 \newcommand*{\setentrycounter}[1]{\def\glsetentrycounter{#1}}

```

The current glossary style can be set using `\glossarystyle{<style>}`.

`\glossarystyle`

```
2581 \newcommand*{\glossarystyle}[1]{%
2582 \ifundefined{glsstyle@#1}{\PackageError{glossaries}{Glossary
2583 style ‘#1’ undefined}{}}{%
2584 \csname @glsstyle@#1\endcsname}}
```

`\newglossarystyle` New glossary styles can be defined using:

```
\newglossarystyle{<name>}{<definition>}
```

The *<definition>* argument should redefine `\theglossary`, `\glossaryheader`, `\glsgroupheading`, `\glossaryentryfield` and `\glsgroupskip` (see [subsection 4.18](#) for the definitions of predefined styles). Glossary styles should not redefine `\glossarypreamble` and `\glossarypostamble`, as the user should be able to switch between styles without affecting the pre- and postambles.

```
2585 \newcommand{\newglossarystyle}[2]{%
2586 \@ifundefined{glsstyle@#1}{%
2587 \expandafter\def\csname @glsstyle@#1\endcsname{#2}}{%
2588 \PackageError{glossaries}{Glossary style ‘#1’ is already defined}{}}
```

Glossary entries are encoded so that the second argument to `\glossaryentryfield` is always specified as `\glsnamefont{<name>}`. This allows the user to change the font used to display the name term without having to redefine `\glossaryentryfield`. The default uses the surrounding font, so in the list type styles (which place the name in the optional argument to `\item`) the name will appear in bold.

`\glsnamefont`

```
2589 \newcommand*{\glsnamefont}[1]{#1}
```

Each glossary entry has an associated number list (usually page numbers) that indicate where in the document the entry has been used. The format for these number lists can be changed using the `format` key in commands like `\glslink`. The default format is given by `\glshypernumber`. This takes a single argument which may be a single number, a number range or a number list. The number ranges are delimited with `\delimR`, the number lists are delimited with `\delimN`.

If the document doesn't have hyperlinks, the numbers can be displayed just as they are, but if the document supports hyperlinks, the numbers should link to the relevant location. This means extracting the individual numbers from the list or ranges. The package does this with the `\hyperpage` command, but this is encoded for comma and dash delimiters and only for the page counter, but this code needs to be more general. So I have adapted the code used in the package.

`\glshypernumber`

```
2590 \@ifundefined{hyperlink}{%
2591 \def\glshypernumber#1{#1}}{%
2592 \def\glshypernumber#1{%
2593 \@glshypernumber#1\nohyperpage{}\@nil}}
```

`\@glshypernumber` This code was provided by Heiko Oberdiek to allow material to be attached to the location.

```

2594 \def\@glshypernumber#1\nohyperpage#2#3\@nil{%
2595   \ifx\\#1\\%
2596   \else
2597     \@delimR#1\delimR\delimR\\%
2598   \fi
2599   \ifx\\#2\\%
2600   \else
2601     #2%
2602   \fi
2603   \ifx\\#3\\%
2604   \else
2605     \@glshypernumber#3\@nil
2606   \fi
2607 }

```

`\@delimR` displays a range of numbers for the counter whose name is given by `\@gls@counter` (which must be set prior to using `\glshypernumber`).

`\@delimR`

```

2608 \def\@delimR#1\delimR #2\delimR #3\\{%
2609 \ifx\\#2\\%
2610   \@delimN{#1}%
2611 \else
2612   \@gls@numberlink{#1}\delimR\@gls@numberlink{#2}%
2613 \fi}

```

`\@delimN` displays a list of individual numbers, instead of a range:

`\@delimN`

```

2614 \def\@delimN#1{\@@delimN#1\delimN \delimN\\}
2615 \def\@@delimN#1\delimN #2\delimN#3\\{%
2616 \ifx\\#3\\%
2617   \@gls@numberlink{#1}%
2618 \else
2619   \@gls@numberlink{#1}\delimN\@gls@numberlink{#2}%
2620 \fi
2621 }

```

The following code is modified from `hyperref's \HyInd@pagelink` where the name of the counter being used is given by `\@gls@counter`.

```

2622 \def\@gls@numberlink#1{%
2623 \begingroup
2624 \toks@={}%
2625 \@gls@removespaces#1 \@nil
2626 \endgroup}
2627 \def\@gls@removespaces#1 #2\@nil{%
2628 \toks@=\expandafter{\the\toks@#1}%

```



```

2629 \ifx\#2\%
2630 \edef\x{\the\toks@}%
2631 \ifx\x\empty
2632 \else
2633 \hyperlink{\glentrycounter.\the\toks@}{\the\toks@}%
2634 \fi
2635 \else
2636 \@gls@ReturnAfterFi{%
2637 \@gls@removespaces#2\@nil
2638 }%
2639 \fi
2640 }
2641 \long\def\@gls@ReturnAfterFi#1\fi{\fi#1}

```

The following commands will switch to the appropriate font, and create a hyperlink, if hyperlinks are supported. If hyperlinks are not supported, they will just display their argument in the appropriate font.

```

\hyperrm
2642 \newcommand*{\hyperrm}[1]{\textrm{\glshypernumber{#1}}}

\hypersf
2643 \newcommand*{\hypersf}[1]{\textsf{\glshypernumber{#1}}}

\hypertt
2644 \newcommand*{\hypertt}[1]{\texttt{\glshypernumber{#1}}}

\hyperbf
2645 \newcommand*{\hyperbf}[1]{\textbf{\glshypernumber{#1}}}

\hypermd
2646 \newcommand*{\hypermd}[1]{\textmd{\glshypernumber{#1}}}

\hyperit
2647 \newcommand*{\hyperit}[1]{\textit{\glshypernumber{#1}}}

\hypersl
2648 \newcommand*{\hypersl}[1]{\textsl{\glshypernumber{#1}}}

\hyperup
2649 \newcommand*{\hyperup}[1]{\textup{\glshypernumber{#1}}}

\hypersc
2650 \newcommand*{\hypersc}[1]{\textsc{\glshypernumber{#1}}}

\hyperemph
2651 \newcommand*{\hyperemph}[1]{\emph{\glshypernumber{#1}}}

```

4.16 Acronyms

If the `acronym` package option is used, a new glossary called `acronym` is created

```
2652 \ifglsacronym
2653   \newglossary[alg]{acronym}{acr}{acn}{\acronymname}
      and \acronymtype is set to the name of this new glossary.
2654   \renewcommand*{\acronymtype}{acronym}
2655 \fi
```

```
\oldacronym \oldacronym[\langle label \rangle]{\langle abbrev \rangle}{\langle long \rangle}{\langle key-val list \rangle}
```

This emulates the way the old package defined acronyms. It is equivalent to `\newacronym[\langle key-val list \rangle]{\langle label \rangle}{\langle abbrev \rangle}{\langle long \rangle}` and it additionally defines the command `\langle label \rangle` which is equivalent to `\gls{\langle label \rangle}` (thus `\langle label \rangle` must only contain alphabetical characters). If `\langle label \rangle` is omitted, `\langle abbrev \rangle` is used. This only emulates the syntax of the old package. The way the acronyms appear in the list of acronyms is determined by the definition of `\newacronym` and the glossary style.

Note that `\langle label \rangle` can't have an optional argument if the package is loaded. If hasn't been loaded then you can do `\langle label \rangle[\langle insert \rangle]` but you can't do `\langle label \rangle[\langle key-val list \rangle]`. For example if you define the acronym `svm`, then you can do `\svm['s]` but you can't do `\svm[format=textbf]`. If the package is loaded, `\svm['s]` will appear as `svm ['s]` which is unlikely to be the desired result. In this case, you will need to use `\gls` explicitly, e.g. `\gls{svm}['s]`. Note that it is up to the user to load if desired.

```
2656 \newcommand{\oldacronym}[4][\gls@label]{%
2657   \def\gls@label{#2}%
2658   \newacronym[#4]{#1}{#2}{#3}%
2659   \@ifundefined{xspace}{%
2660     \expandafter\edef\csname#1\endcsname{%
2661       \noexpand\@ifstar{\noexpand\Gls{#1}}{\noexpand\gls{#1}}}%
2662   }{%
2663     \expandafter\edef\csname#1\endcsname{%
2664       \noexpand\@ifstar{\noexpand\Gls{#1}\noexpand\xspace}{%
2665         \noexpand\gls{#1}\noexpand\xspace}%
2666     }%
2667 }
```

```
\newacronym[\langle key-val list \rangle]{\langle label \rangle}{\langle abbrev \rangle}{\langle long \rangle}
```

This is a quick way of defining acronyms, all it does is call `\newglossaryentry` with the appropriate values. It sets the glossary type to `\acronymtype` which will be `acronym` if the package option `acronym` has been used, otherwise it will be the default glossary. Since `\newacronym` merely calls `\newglossaryentry`, the acronym is treated like any other glossary entry.

If you prefer a different format, you can redefine `\newacronym` as required. The optional argument can be used to override any of the settings.

This is just a stub. It's redefined by commands like `\SetDefaultAcronymStyle`.

`\newacronym`

2668 `\newcommand{\newacronym}[4] [] {}`

Set up some convenient short cuts. These need to be changed if `\newacronym` is changed (or if the `description` key is changed).

`\acrpluralsuffix` Plural suffix used by `\newacronym`. This just defaults to `\glspluralsuffix` but is changed to include `\textup` if the smallcaps option is used, so that the suffix doesn't appear in small caps as it doesn't look right. For example, ABCS looks as though the "s" is part of the acronym, but ABCs looks as though the "s" is a plural suffix. Since the entire text `abcs` is set in `\textsc`, `\textup` is needed to cancel it out.

2669 `\newcommand*{\acrpluralsuffix}{\glspluralsuffix}`

Make a note of the keys that are used to store the long and short forms:

`\glsshortkey`

2670 `\newcommand*{\glsshortkey}{text}`

`\glsshortpluralkey`

2671 `\newcommand*{\glsshortpluralkey}{plural}`

`\glslongkey`

2672 `\newcommand*{\glslongkey}{description}`

`\glslongpluralkey`

2673 `\newcommand*{\glslongpluralkey}{descriptionplural}`

Using the default definitions, `\acrshort` is the same as `\glstext`, which means that it will print the abbreviation.

`\acrshort`

2674 `\newcommand*{\acrshort}[2] [] {}`

2675 `\new@ifnextchar[{\@acrshort{#1}{#2}}{\@acrshort{#1}{#2} [] {}}`

`\Acrshort`

2676 `\newcommand*{\Acrshort}[2] [] {}`

2677 `\new@ifnextchar[{\@Acrshort{#1}{#2}}{\@Acrshort{#1}{#2} [] {}}`

`\ACRshort`

2678 `\newcommand*{\ACRshort}[2] [] {}`

2679 `\new@ifnextchar[{\@ACRshort{#1}{#2}}{\@ACRshort{#1}{#2} [] {}}`

Plural:

`\acrshortpl`

2680 `\newcommand*{\acrshortpl}[2] [] {}`

2681 `\new@ifnextchar[{\@acrshortpl{#1}{#2}}{\@acrshortpl{#1}{#2} [] {}}`

```

\Acrshortpl
2682 \newcommand*\Acrshortpl}[2] [] {%
2683   \new@ifnextchar[{\@Acrshortpl{#1}{#2}}{\@Acrshortpl{#1}{#2} []}]

\ACRshortpl
2684 \newcommand*\ACRshortpl}[2] [] {%
2685   \new@ifnextchar[{\@ACRshortpl{#1}{#2}}{\@ACRshortpl{#1}{#2} []}]

    \acrlong is set to \glsdesc, so it will print the long form, unless the descrip-
    tion key has been set to something else.

\acrlong
2686 \newcommand*\acrlong}[2] [] {%
2687   \new@ifnextchar[{\@acrlong{#1}{#2}}{\@acrlong{#1}{#2} []}]

\Acrlong
2688 \newcommand*\Acrlong}[2] [] {%
2689   \new@ifnextchar[{\@Acrlong{#1}{#2}}{\@Acrlong{#1}{#2} []}]

\ACRlong
2690 \newcommand*\ACRlong}[2] [] {%
2691   \new@ifnextchar[{\@ACRlong{#1}{#2}}{\@ACRlong{#1}{#2} []}]

    Plural:

\acrlongpl
2692 \newcommand*\acrlongpl}[2] [] {%
2693   \new@ifnextchar[{\@acrlongpl{#1}{#2}}{\@acrlongpl{#1}{#2} []}]

\Acrlongpl
2694 \newcommand*\Acrlongpl}[2] [] {%
2695   \new@ifnextchar[{\@Acrlongpl{#1}{#2}}{\@Acrlongpl{#1}{#2} []}]

\ACRlongpl
2696 \newcommand*\ACRlongpl}[2] [] {%
2697   \new@ifnextchar[{\@ACRlongpl{#1}{#2}}{\@ACRlongpl{#1}{#2} []}]

    \acrfull is set to \glsfirst, so it should display the full form.

\acrfull
2698 \newcommand*\acrfull}[2] [] {%
2699   \new@ifnextchar[{\@acrfull{#1}{#2}}{\@acrfull{#1}{#2} []}]

\ACRfull
2700 \newcommand*\ACRfull}[2] [] {%
2701   \new@ifnextchar[{\@ACRfull{#1}{#2}}{\@ACRfull{#1}{#2} []}]

\ACRfull
2702 \newcommand*\ACRfull}[2] [] {%
2703   \new@ifnextchar[{\@ACRfull{#1}{#2}}{\@ACRfull{#1}{#2} []}]

```

Plural:

```

\acrfullpl
2704 \newcommand*{\acrfullpl}[2] [] {%
2705   \new@ifnextchar[{\@acrfullpl{#1}{#2}}]{\@acrfullpl{#1}{#2} []}}

\Acrfullpl
2706 \newcommand*{\Acrfullpl}[2] [] {%
2707   \new@ifnextchar[{\@Acrfullpl{#1}{#2}}]{\@Acrfullpl{#1}{#2} []}}

\ACRfullpl
2708 \newcommand*{\ACRfullpl}[2] [] {%
2709   \new@ifnextchar[{\@ACRfullpl{#1}{#2}}]{\@ACRfullpl{#1}{#2} []}}

```

4.17 Predefined acronym styles

`\acronymfont` This is only used with the additional acronym styles:

```
2710 \newcommand{\acronymfont}[1]{#1}
```

`\firstacronymfont` This is only used with the additional acronym styles:

```
2711 \newcommand{\firstacronymfont}[1]{\acronymfont{#1}}
```

`\acrnameformat` The styles that allow an additional description use `\acrnameformat{<short>}{<long>}` to determine what information is displayed in the name.

```
2712 \newcommand*{\acrnameformat}[2]{\acronymfont{#1}}
```

Define some tokens used by `\newacronym`:

```

\glskeylisttok
2713 \newtoks\glskeylisttok

```

```

\glslabeltok
2714 \newtoks\glslabeltok

```

```

\glsshorttok
2715 \newtoks\glsshorttok

```

```

\glslongtok
2716 \newtoks\glslongtok

```

`\newacronymhook` Provide a hook for `\newacronym`:

```
2717 \newcommand*{\newacronymhook}{}
```

`\SetDefaultAcronymDisplayStyle` Sets the default acronym display style for given glossary.

```

2718 \newcommand*{\SetDefaultAcronymDisplayStyle}[1] {%
2719   \defglsdisplay[#1]{##1##4}%
2720   \defglsdisplayfirst[#1]{##1##4}%
2721 }

```

`\DefaultNewAcronymDef` Sets up the acronym definition for the default style. The information is provided by the tokens `\glslabeltok`, `\glsshorttok`, `\glslongtok` and `\glskeylisttok`.

```

2722 \newcommand*{\DefaultNewAcronymDef}{%
2723   \edef\@do@newglossaryentry{%
2724     \noexpand\newglossaryentry{\the\glslabeltok}%
2725     {%
2726       type=\acronymtype,%
2727       name={\the\glsshorttok},%
2728       description={\the\glslongtok},%
2729       text={\the\glsshorttok},%
2730       sort={\the\glsshorttok},%
2731       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
2732       first={\the\glslongtok\space(\the\glsshorttok)},%
2733       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
2734       firstplural={\noexpand\@glo@descplural\space
2735         (\noexpand\@glo@plural)},%
2736       \the\glskeylisttok
2737     }%
2738   }%
2739   \@do@newglossaryentry
2740 }

```

`\SetDefaultAcronymStyle` Set up the default acronym style:

```

2741 \newcommand*{\SetDefaultAcronymStyle}{%

```

Set the display style:

```

2742   \@for\@gls@type:=\@glsacronymlists\do{%
2743     \SetDefaultAcronymDisplayStyle{\@gls@type}%
2744   }%

```

Set up the definition of `\newacronym`:

```

2745   \renewcommand{\newacronym}[4][{}]{%

```

If user is just using the main glossary and hasn't identified it as a list of acronyms, then update. (This is done to ensure backwards compatibility with versions prior to 2.04).

```

2746     \ifx\@glsacronymlists\@empty
2747       \def\@glo@type{\acronymtype}%
2748       \setkeys{glossentry}{##1}%
2749       \DeclareAcronymList{\@glo@type}%
2750       \SetDefaultAcronymDisplayStyle{\@glo@type}%
2751     \fi
2752     \glskeylisttok{##1}%
2753     \glslabeltok{##2}%
2754     \glsshorttok{##3}%
2755     \glslongtok{##4}%
2756     \newacronymhook
2757     \DefaultNewAcronymDef
2758   }%

```

Define short cuts.

```

2759 \renewcommand*{\acrpluralsuffix}{\glspluralsuffix}%
2760 \renewcommand*{\glsshortkey}{\text}%
2761 \renewcommand*{\glsshortpluralkey}{\plural}%
2762 \renewcommand*{\glslongkey}{\description}%
2763 \renewcommand*{\glslongpluralkey}{\descriptionplural}%
2764 \def\@acrshort##1##2[##3]{\@glstext@{##1}{##2}[##3]}%
2765 \def\@Acrshort##1##2[##3]{\@Glstext@{##1}{##2}[##3]}%
2766 \def\@ACRshort##1##2[##3]{\@GLStext@{##1}{##2}[##3]}%
2767 \def\@acrshortpl##1##2[##3]{\@glsplural@{##1}{##2}[##3]}%
2768 \def\@Acrshortpl##1##2[##3]{\@Glsplural@{##1}{##2}[##3]}%
2769 \def\@ACRshortpl##1##2[##3]{\@GLSplural@{##1}{##2}[##3]}%
2770 \def\@acrlong##1##2[##3]{\@glsdesc@{##1}{##2}[##3]}%
2771 \def\@Acrlong##1##2[##3]{\@Glsdesc@{##1}{##2}[##3]}%
2772 \def\@ACRlong##1##2[##3]{\@GLSdesc@{##1}{##2}[##3]}%
2773 \def\@acrlongpl##1##2[##3]{\@glsdescplural@{##1}{##2}[##3]}%
2774 \def\@Acrlongpl##1##2[##3]{\@Glsdescplural@{##1}{##2}[##3]}%
2775 \def\@ACRlongpl##1##2[##3]{\@GLSdescplural@{##1}{##2}[##3]}%
2776 \def\@acrfull##1##2[##3]{\@glsfirst@{##1}{##2}[##3]}%
2777 \def\@Acrfull##1##2[##3]{\@Glsfirst@{##1}{##2}[##3]}%
2778 \def\@ACRfull##1##2[##3]{\@GLSfirst@{##1}{##2}[##3]}%
2779 \def\@acrfullpl##1##2[##3]{\@glsfirstplural@{##1}{##2}[##3]}%
2780 \def\@Acrfullpl##1##2[##3]{\@Glsfirstplural@{##1}{##2}[##3]}%
2781 \def\@ACRfullpl##1##2[##3]{\@GLSfirstplural@{##1}{##2}[##3]}%
2782 }

```

tionFootnoteAcronymDisplayStyle Sets the acronym display style for given glossary for the description and footnote combination.

```

2783 \newcommand*{\SetDescriptionFootnoteAcronymDisplayStyle}[1]{%
2784   \defglsdisplayfirst[#1]{%
2785     \firstacronymfont{##1}##4%
2786     \protect\footnote{%
2787       \glslink[\@gls@link@opts]{\@gls@link@label}{##3}}}%
2788   \defglsdisplay[#1]{\acronymfont{##1}##4}%
2789 }

```

escriptionFootnoteNewAcronymDef

```

2790 \newcommand*{\DescriptionFootnoteNewAcronymDef}{%
2791   \edef\@do@newglossaryentry{%
2792     \noexpand\newglossaryentry{\the\glslabeltok}%
2793     {%
2794       type=\acronymtype,%
2795       name={\noexpand\acronymfont{\the\glsshorttok}},%
2796       sort={\the\glsshorttok},%
2797       text={\the\glsshorttok},%
2798       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
2799       symbol={\the\glslongtok},%
2800       symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
2801       \the\glskeylisttok

```

```

2802     }%
2803     }%
2804     \@do@newglossaryentry
2805 }

```

DescriptionFootnoteAcronymStyle If a description and footnote are both required, store the long form in the **symbol** key. Store the short form in **text** key. Note that since the long form is stored in the symbol key, if you want the long form to appear in the list of acronyms, you need to use a glossary style that displays the symbol key.

```

2806 \newcommand*{\SetDescriptionFootnoteAcronymStyle}{%
2807   \renewcommand{\newacronym}[4][]{%
2808     \ifx\@glsacronymlists\@empty
2809       \def\@glo@type{\acronymtype}%
2810       \setkeys{glossentry}{##1}%
2811       \DeclareAcronymList{\@glo@type}%
2812       \SetDescriptionFootnoteAcronymDisplayStyle{\@glo@type}%
2813     \fi
2814     \glskeylisttok{##1}%
2815     \glslabeltok{##2}%
2816     \glsshorttok{##3}%
2817     \glslongtok{##4}%
2818     \newacronymhook
2819     \DescriptionFootnoteNewAcronymDef
2820   }%

```

Set up the commands to make a note of the keys to store the long and short forms:

```

2821 \def\glsshortkey{text}%
2822 \def\glsshortpluralkey{plural}%
2823 \def\glslongkey{symbol}%
2824 \def\glslongpluralkey{symbolplural}%

```

Set up short cuts. Short form:

```

2825 \def\@acrshort##1##2[##3]{%
2826   \acronymfont{\@glstext@{##1}{##2}[##3]}}%
2827 \def\@Acrshort##1##2[##3]{%
2828   \acronymfont{\@Glstext@{##1}{##2}[##3]}}%
2829 \def\@ACRshort##1##2[##3]{%
2830   \acronymfont{\@GLStext@{##1}{##2}[##3]}}%

```

Plural form:

```

2831 \def\@acrshortpl##1##2[##3]{%
2832   \acronymfont{\@glsplural@{##1}{##2}[##3]}}%
2833 \def\@Acrshortpl##1##2[##3]{%
2834   \acronymfont{\@Glsplural@{##1}{##2}[##3]}}%
2835 \def\@ACRshortpl##1##2[##3]{%
2836   \acronymfont{\@GLSplural@{##1}{##2}[##3]}}%

```

Long form:

```

2837 \def\@acrlong##1##2[##3]{\@glssymbol@{##1}{##2}[##3]}}%
2838 \def\@Acrlong##1##2[##3]{\@Glsymbol@{##1}{##2}[##3]}}%
2839 \def\@ACRlong##1##2[##3]{\@GLSsymbol@{##1}{##2}[##3]}}%

```


Plural long form:

```
2840 \def\@acrlongpl##1##2[##3]{\@glssymbolplural@{##1}-{##2}[##3]}%
2841 \def\@Acrlongpl##1##2[##3]{\@Glssymbolplural@{##1}-{##2}[##3]}%
2842 \def\@ACRlongpl##1##2[##3]{\@GLSsymbolplural@{##1}-{##2}[##3]}%
```

Full form:

```
2843 \def\@acrfull##1##2[##3]{\@glssymbol@{##1}-{##2}[##3]
2844   (\acronymfont{\@glstext@{##1}-{##2}[##3]})}%
2845 \def\@Acrfull##1##2[##3]{\@Glssymbol@{##1}-{##2}[##3]
2846   (\acronymfont{\@glstext@{##1}-{##2}[##3]})}%
2847 \def\@ACRfull##1##2[##3]{\@GLSsymbol@{##1}-{##2}[##3]
2848   (\acronymfont{\@GLStext@{##1}-{##2}[##3]})}%
```

Plural full form:

```
2849 \def\@acrfullpl##1##2[##3]{\@glssymbolplural@{##1}-{##2}[##3]
2850   (\acronymfont{\@glsplural@{##1}-{##2}[##3]})}%
2851 \def\@Acrfullpl##1##2[##3]{\@Glssymbolplural@{##1}-{##2}[##3]
2852   (\acronymfont{\@glsplural@{##1}-{##2}[##3]})}%
2853 \def\@ACRfullpl##1##2[##3]{\@GLSsymbolplural@{##1}-{##2}[##3]
2854   (\acronymfont{\@GLSplural@{##1}-{##2}[##3]})}%
```

If footnote package option is specified, set the first use to append the long form (stored in symbol) as a footnote.

```
2855 \@for\@gls@type:=\@glsacronymlists\do{%
2856   \SetDescriptionFootnoteAcronymDisplayStyle{\@gls@type}%
2857 }%
```

Redefine `\acronymfont` if small caps required. The plural suffix is set in an upright font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```
2858 \ifglssacrsmallcaps
2859   \renewcommand*\acronymfont[1]{\textsc{##1}}%
2860   \renewcommand*\acrpluralsuffix{%
2861     \textup{\glspluralsuffix}}%
2862 \else
2863   \ifglssacrsmaller
2864     \renewcommand*\acronymfont[1]{\textsmaller{##1}}%
2865   \fi
2866 \fi
```

Check for package option clash

```
2867 \ifglssacrdua
2868   \PackageError{glossaries}{Option clash: ‘footnote’ and ‘dua’
2869     can’t both be set}{}%
2870 \fi
2871 }%
```

`\SetDescriptionDUAAcronymDisplayStyle` Sets the acronym display style for given glossary with description and dua combination.

```
2872 \newcommand*\SetDescriptionDUAAcronymDisplayStyle[1]{%
2873   \defglstdisplay[#1]{##1##4}%
```

```

2874 \defglsdisplayfirst[#1]{##1##4}%
2875 }

```

`\DescriptionDUANewAcronymDef`

```

2876 \newcommand*{\DescriptionDUANewAcronymDef}{%
2877 \edef\@do@newglossaryentry{%
2878 \noexpand\newglossaryentry{\the\glslabeltok}%
2879 {%
2880 type=\acronymtype,%
2881 name={\the\glslongtok},%
2882 sort={\the\glslongtok},
2883 text={\the\glslongtok},%
2884 plural={\the\glslongtok\noexpand\acrpluralsuffix},%
2885 symbol={\the\glsshorttok},%
2886 symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
2887 \the\glskeylisttok
2888 }%
2889 }%
2890 \@do@newglossaryentry
2891 }

```

`\SetDescriptionDUAAcronymStyle` Description, don't use acronym and no footnote. Note that the short form is stored in the symbol key, so if the short form needs to be displayed in the glossary, use a style the displays the symbol.

```

2892 \newcommand*{\SetDescriptionDUAAcronymStyle}{%
2893 \ifglsacrsmallcaps
2894 \PackageError{glossaries}{Option clash: 'smallcaps' and 'dua'
2895 can't both be set}{}%
2896 \else
2897 \ifglsacrsmaller
2898 \PackageError{glossaries}{Option clash: 'smaller' and 'dua'
2899 can't both be set}{}%
2900 \fi
2901 \fi
2902 \renewcommand{\newacronym}[4][ ]{%
2903 \ifx\@glsacronymlists\@empty
2904 \def\@glo@type{\acronymtype}%
2905 \setkeys{glossentry}{##1}%
2906 \DeclareAcronymList{\@glo@type}%
2907 \SetDescriptionDUAAcronymDisplayStyle{\@glo@type}%
2908 \fi
2909 \glskeylisttok{##1}%
2910 \glslabeltok{##2}%
2911 \glsshorttok{##3}%
2912 \glslongtok{##4}%
2913 \newacronymhook
2914 \DescriptionDUANewAcronymDef
2915 }%

```

Set up the commands to make a note of the keys to store the long and short forms:

```

2916 \def\glsshortkey{symbol}%
2917 \def\glsshortpluralkey{symbolplural}%
2918 \def\glslongkey{first}%
2919 \def\glslongpluralkey{plural}%

Set up short cuts. Short form:
2920 \def\@acrshort##1##2[##3]{%
2921   \acronymfont{\@glssymbol@{##1}{##2}[##3]}}%
2922 \def\@Acrshort##1##2[##3]{%
2923   \acronymfont{\@Glsymbol@{##1}{##2}[##3]}}%
2924 \def\@ACRshort##1##2[##3]{%
2925   \acronymfont{\@GLSsymbol@{##1}{##2}[##3]}}%

Plural short form:
2926 \def\@acrshortpl##1##2[##3]{%
2927   \acronymfont{\@glssymbolplural@{##1}{##2}[##3]}}%
2928 \def\@Acrshortpl##1##2[##3]{%
2929   \acronymfont{\@Glsymbolplural@{##1}{##2}[##3]}}%
2930 \def\@ACRshortpl##1##2[##3]{%
2931   \acronymfont{\@GLSsymbolplural@{##1}{##2}[##3]}}%

Long form:
2932 \def\@acrlong##1##2[##3]{\@glsfirst@{##1}{##2}[##3]}%
2933 \def\@Acrlong##1##2[##3]{\@Glsfirst@{##1}{##2}[##3]}%
2934 \def\@ACRlong##1##2[##3]{\@GLSfirst@{##1}{##2}[##3]}%

Plural long form:
2935 \def\@acrlongpl##1##2[##3]{\@glsfirstplural@{##1}{##2}[##3]}%
2936 \def\@Acrlongpl##1##2[##3]{\@Glsfirstplural@{##1}{##2}[##3]}%
2937 \def\@ACRlongpl##1##2[##3]{\@GLSfirstplural@{##1}{##2}[##3]}%

Full form:
2938 \def\@acrfull##1##2[##3]{\@glsfirst@{##1}{##2}[##3]
2939   (\acronymfont{\@glssymbol@{##1}{##2}[##3]})}%
2940 \def\@Acrfull##1##2[##3]{\@Glsfirst@{##1}{##2}[##3]
2941   (\acronymfont{\@Glsymbol@{##1}{##2}[##3]})}%
2942 \def\@ACRfull##1##2[##3]{\@GLSfirst@{##1}{##2}[##3]
2943   (\acronymfont{\@GLSsymbol@{##1}{##2}[##3]})}%

Plural full form:
2944 \def\@acrfullpl##1##2[##3]{\@glsfirstplural@{##1}{##2}[##3]
2945   (\acronymfont{\@glssymbolplural@{##1}{##2}[##3]})}%
2946 \def\@Acrfullpl##1##2[##3]{\@Glsfirstplural@{##1}{##2}[##3]
2947   (\acronymfont{\@Glsymbolplural@{##1}{##2}[##3]})}%
2948 \def\@ACRfullpl##1##2[##3]{\@GLSfirstplural@{##1}{##2}[##3]
2949   (\acronymfont{\@GLSsymbolplural@{##1}{##2}[##3]})}%

Set display.
2950 \@for\@gls@type:=\@glsacronymlists\do{%
2951   \SetDescriptionDUAAcronymDisplayStyle{\@gls@type}%
2952 }%
2953 }%

```

`\DescriptionAcronymDisplayStyle` Sets the acronym display style for given glossary using the description setting (but not footnote or dua).

```

2954 \newcommand*{\SetDescriptionAcronymDisplayStyle}[1]{%
2955   \defglsdisplayfirst[#1]{%
2956     ##1##4 (\firstacronymfont{##3})}%
2957   \defglsdisplay[#1]{\acronymfont{##1}##4}%
2958 }

```

`\DescriptionNewAcronymDef`

```

2959 \newcommand*{\DescriptionNewAcronymDef}{%
2960   \edef\@do@newglossaryentry{%
2961     \noexpand\newglossaryentry{\the\glslabeltok}%
2962     {%
2963       type=\acronymtype,%
2964       name={\noexpand
2965         \acronymformat{\the\glsshorttok}{\the\glslongtok}},%
2966       sort={\the\glsshorttok},%
2967       first={\the\glslongtok},%
2968       firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
2969       text={\the\glsshorttok},%
2970       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
2971       symbol={\noexpand\@glo@text},%
2972       symbolplural={\noexpand\@glo@plural},%
2973       \the\glskeylisttok}%
2974   }%
2975   \@do@newglossaryentry
2976 }

```

`\SetDescriptionAcronymStyle` Option description is used, but not dua or footnote. Store long form in first key and short form in text and symbol key. The name is stored using `\acronymformat` to allow the user to override the way the name is displayed in the list of acronyms.

```

2977 \newcommand*{\SetDescriptionAcronymStyle}{%
2978   \renewcommand{\newacronym}[4][\]{%
2979     \ifx\@glsacronymlists\@empty
2980       \def\@glo@type{\acronymtype}%
2981       \setkeys{glossentry}{##1}%
2982       \DeclareAcronymList{\@glo@type}%
2983       \SetDescriptionAcronymDisplayStyle{\@glo@type}%
2984     \fi
2985     \glskeylisttok{##1}%
2986     \glslabeltok{##2}%
2987     \glsshorttok{##3}%
2988     \glslongtok{##4}%
2989     \newacronymhook
2990     \DescriptionNewAcronymDef
2991   }%

```

Set up the commands to make a note of the keys to store the long and short forms:

```

2992 \def\glsshortkey{text}%

```

```

2993 \def\glsshortpluralkey{plural}%
2994 \def\glslongkey{first}%
2995 \def\glslongpluralkey{firstplural}%

Set up short cuts. Short form:
2996 \def\@acrshort##1##2[##3]{%
2997   \acronymfont{\@glstext@{##1}-{##2}[##3]}}%
2998 \def\@Acrshort##1##2[##3]{%
2999   \acronymfont{\@Glstext@{##1}-{##2}[##3]}}%
3000 \def\@ACRshort##1##2[##3]{%
3001   \acronymfont{\@GLStext@{##1}-{##2}[##3]}}%

Plural short form:
3002 \def\@acrshortpl##1##2[##3]{%
3003   \acronymfont{\@glsplural@{##1}-{##2}[##3]}}%
3004 \def\@Acrshortpl##1##2[##3]{%
3005   \acronymfont{\@Glsplural@{##1}-{##2}[##3]}}%
3006 \def\@ACRshortpl##1##2[##3]{%
3007   \acronymfont{\@GLSplural@{##1}-{##2}[##3]}}%

Long form:
3008 \def\@acrlong##1##2[##3]{\@glsfirst@{##1}-{##2}[##3]}}%
3009 \def\@Acrlong##1##2[##3]{\@Glsfirst@{##1}-{##2}[##3]}}%
3010 \def\@ACRlong##1##2[##3]{\@GLSfirst@{##1}-{##2}[##3]}}%

Plural long form:
3011 \def\@acrlongpl##1##2[##3]{\@glsfirstplural@{##1}-{##2}[##3]}}%
3012 \def\@Acrlongpl##1##2[##3]{\@Glsfirstplural@{##1}-{##2}[##3]}}%
3013 \def\@ACRlongpl##1##2[##3]{\@GLSfirstplural@{##1}-{##2}[##3]}}%

Full form:
3014 \def\@acrfull##1##2[##3]{\@glsfirst@{##1}-{##2}[##3]
3015   (\acronymfont{\@glssymbol@{##1}-{##2}[##3]})}%
3016 \def\@Acrfull##1##2[##3]{\@Glsfirst@{##1}-{##2}[##3]
3017   (\acronymfont{\@glssymbol@{##1}-{##2}[##3]})}%
3018 \def\@ACRfull##1##2[##3]{\@GLSfirst@{##1}-{##2}[##3]
3019   (\acronymfont{\@GLSsymbol@{##1}-{##2}[##3]})}%

Plural full form:
3020 \def\@acrfullpl##1##2[##3]{\@glsfirstplural@{##1}-{##2}[##3]
3021   (\acronymfont{\@glssymbolplural@{##1}-{##2}[##3]})}%
3022 \def\@Acrfullpl##1##2[##3]{\@Glsfirstplural@{##1}-{##2}[##3]
3023   (\acronymfont{\@glssymbolplural@{##1}-{##2}[##3]})}%
3024 \def\@ACRfullpl##1##2[##3]{\@GLSfirstplural@{##1}-{##2}[##3]
3025   (\acronymfont{\@GLSsymbolplural@{##1}-{##2}[##3]})}%

Set display.
3026 \@for\@gls@type:=\@glsacronymlists\do{%
3027   \SetDescriptionAcronymDisplayStyle{\@gls@type}%
3028 }%

```

Redefine `\acronymfont` if small caps required. The plural suffix is set in an upright font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```

3029 \ifglsmallcaps
3030   \renewcommand{\acronymfont}[1]{\textsc{##1}}
3031   \renewcommand*{\acrpluralsuffix}{%
3032     \textup{\glpluralsuffix}}%
3033   \else
3034     \ifglsmaller
3035       \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
3036     \fi
3037   \fi
3038 }%
```

`\SetFootnoteAcronymDisplayStyle` Sets the acronym display style for given glossary with footnote setting (but not description or dua).

```

3039 \newcommand*{\SetFootnoteAcronymDisplayStyle}[1]{%
3040   \defgldisplayfirst[#1]{%
3041     \firstacronymfont{##1}##4\protect\footnote{%
3042       \protect\glslink
3043         [\@gls@link@opts]{\@gls@link@label}{##2}}}%
3044   \defgldisplay[#1]{\acronymfont{##1}##4}%
3045 }
```

`\FootnoteNewAcronymDef`

```

3046 \newcommand*{\FootnoteNewAcronymDef}{%
3047   \edef\@do@newglossaryentry{%
3048     \noexpand\newglossaryentry{\the\glslabelltok}%
3049     {%
3050       type=\acronymtype,%
3051       name={\noexpand\acronymfont{\the\glsshorttok}},%
3052       sort={\the\glsshorttok},%
3053       text={\the\glsshorttok},%
3054       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
3055       description={\the\glslongtok},%
3056       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
3057       \the\glskeylisttok
3058     }%
3059   }%
3060   \@do@newglossaryentry
3061 }
```

`\SetFootnoteAcronymStyle` If footnote package option is specified, set the first use to append the long form (stored in description) as a footnote. Use the description key to store the long form.

```

3062 \newcommand*{\SetFootnoteAcronymStyle}{%
3063   \renewcommand{\newacronym}[4][1]{%
3064     \ifx\@glsacronymlists\empty
3065       \def\@glo@type{\acronymtype}%
3066       \setkeys{glossentry}{##1}%
3067     \fi
3068   }%
```

```

3067     \DeclareAcronymList{\@glo@type}%
3068     \SetFootnoteAcronymDisplayStyle{\@glo@type}%
3069     \fi
3070     \glskeylisttok{##1}%
3071     \glslabeltok{##2}%
3072     \glsshorttok{##3}%
3073     \glslongtok{##4}%
3074     \newacronymhook
3075     \FootnoteNewAcronymDef
3076 }%

```

Set up the commands to make a note of the keys to store the long and short forms:

```

3077 \def\glsshortkey{text}%
3078 \def\glsshortpluralkey{plural}%
3079 \def\glslongkey{description}%
3080 \def\glslongpluralkey{descriptionplural}%

```

Set display

```

3081 \@for\@gls@type:=\@glsacronymlists\do{%
3082     \SetFootnoteAcronymDisplayStyle{\@gls@type}%
3083 }%

```

Set up short cuts. Short form:

```

3084 \def\@acrshort##1##2[##3]{\acronymfont{\@glstext@{##1}-{##2}[##3]}}%
3085 \def\@Acrshort##1##2[##3]{\acronymfont{\@Glstext@{##1}-{##2}[##3]}}%
3086 \def\@ACRshort##1##2[##3]{\acronymfont{\@GLStext@{##1}-{##2}[##3]}}%

```

Plural short form:

```

3087 \def\@acrshortpl##1##2[##3]{%
3088     \acronymfont{\@glsplural@{##1}-{##2}[##3]}}%
3089 \def\@Acrshortpl##1##2[##3]{%
3090     \acronymfont{\@Glsplural@{##1}-{##2}[##3]}}%
3091 \def\@ACRshortpl##1##2[##3]{%
3092     \acronymfont{\@GLSplural@{##1}-{##2}[##3]}}%

```

Long form:

```

3093 \def\@acrlong##1##2[##3]{\@glsdesc@{##1}-{##2}[##3]}}%
3094 \def\@Acrlong##1##2[##3]{\@Glsdesc@{##1}-{##2}[##3]}}%
3095 \def\@ACRlong##1##2[##3]{\@GLSdesc@{##1}-{##2}[##3]}}%

```

Plural long form:

```

3096 \def\@acrlongpl##1##2[##3]{\@glsdescplural@{##1}-{##2}[##3]}}%
3097 \def\@Acrlongpl##1##2[##3]{\@Glsdescplural@{##1}-{##2}[##3]}}%
3098 \def\@ACRlongpl##1##2[##3]{\@GLSdescplural@{##1}-{##2}[##3]}}%

```

Full form:

```

3099 \def\@acrfull##1##2[##3]{\@glsdesc@{##1}-{##2}[##3]
3100     (\@glstext@{##1}-{##2}[##3])}%
3101 \def\@Acrfull##1##2[##3]{\@Glsdesc@{##1}-{##2}[##3]
3102     (\@glstext@{##1}-{##2}[##3])}%
3103 \def\@ACRfull##1##2[##3]{\@GLSdesc@{##1}-{##2}[##3]
3104     (\@GLStext@{##1}-{##2}[##3])}%

```

Plural full form:

```

3105 \def\@acrfullpl##1##2[##3]{\@glsdescplural@{##1}-{##2}[##3]
3106 (\@glsplural@{##1}-{##2}[##3])}%
3107 \def\@Acrfullpl##1##2[##3]{\@Glsdesctext@{##1}-{##2}[##3]
3108 (\@glsplural@{##1}-{##2}[##3])}%
3109 \def\@ACRfullpl##1##2[##3]{\@GLSdesctext@{##1}-{##2}[##3]
3110 (\@GLSplural@{##1}-{##2}[##3])}%

```

Redefine `\acronymfont` if small caps required. The plural suffix is set in an upright font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```

3111 \ifglsacrsmallcaps
3112 \renewcommand*{\acronymfont}[1]{\textsc{##1}}%
3113 \renewcommand*{\acrpluralsuffix}{%
3114 \textup{\glspluralsuffix}}%
3115 \else
3116 \ifglsacrsmaller
3117 \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}%
3118 \fi
3119 \fi

```

Check for option clash

```

3120 \ifglsacrdua
3121 \PackageError{glossaries}{Option clash: 'footnote' and 'dua'
3122 can't both be set}{}%
3123 \fi
3124 }%

```

`\SetSmallAcronymDisplayStyle` Sets the acronym display style for given glossary where neither footnote nor description is required, but smallcaps or smaller specified.

```

3125 \newcommand*{\SetSmallAcronymDisplayStyle}[1]{%
3126 \defglsdisplayfirst[#1]{##1##4 (\firstacronymfont{##3})}
3127 \defglsdisplay[#1]{\acronymfont{##1}##4}%
3128 }

```

`\SmallNewAcronymDef`

```

3129 \newcommand*{\SmallNewAcronymDef}{%
3130 \edef\@do@newglossaryentry{%
3131 \noexpand\newglossaryentry{\the\glslabeltok}%
3132 {%
3133 type=\acronymtype,%
3134 name={\noexpand\acronymfont{\the\glsshorttok}},%
3135 sort={\the\glsshorttok},%
3136 text={\noexpand\@glo@symbol},%
3137 plural={\noexpand\@glo@symbolplural},%
3138 first={\the\glslongtok},%
3139 firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
3140 description={\noexpand\@glo@first},%
3141 descriptionplural={\noexpand\@glo@firstplural},%
3142 symbol={\the\glsshorttok},%

```



```

3143     symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
3144     \the\glskeylisttok
3145   }%
3146 }%
3147 \@do@newglossaryentry
3148 }

```

`\SetSmallAcronymStyle` Neither footnote nor description required, but smallcaps or smaller specified. Use the symbol key to store the short form and first to store the long form.

```

3149 \newcommand*{\SetSmallAcronymStyle}{%
3150   \renewcommand{\newacronym}[4][]{%
3151     \ifx\@glsacronymlists\@empty
3152       \def\@gls@type{\acronymtype}%
3153       \setkeys{glossentry}{##1}%
3154       \DeclareAcronymList{\@gls@type}%
3155       \SetSmallAcronymDisplayStyle{\@gls@type}%
3156     \fi
3157     \glskeylisttok{##1}%
3158     \glslabeltok{##2}%
3159     \glsshorttok{##3}%
3160     \glslongtok{##4}%
3161     \newacronymhook
3162     \SmallNewAcronymDef
3163   }%

```

Set up the commands to make a note of the keys to store the long and short forms:

```

3164   \def\glsshortkey{symbol}%
3165   \def\glsshortpluralkey{symbolplural}%
3166   \def\glslongkey{first}%
3167   \def\glslongpluralkey{firstplural}%

```

Change the display since first only contains long form.

```

3168   \@for\@gls@type:=\@glsacronymlists\do{%
3169     \SetSmallAcronymDisplayStyle{\@gls@type}%
3170   }%

```

Redefine `\acronymfont` if small caps required. The plural suffix is set in an upright font so that it remains in normal lower case, otherwise it looks as though it's part of the acronym.

```

3171   \ifglscrsmallcaps
3172     \renewcommand*{\acronymfont}[1]{\textsc{##1}}
3173     \renewcommand*{\acrpluralsuffix}{%
3174       \textup{\glspluralsuffix}}%
3175   \else
3176     \renewcommand*{\acronymfont}[1]{\textsmaller{##1}}
3177   \fi

```

Set up short cuts. Short form:

```

3178   \def\@acrshort##1##2[##3]{%
3179     \acronymfont{\@gls@text@{##1}{##2}[##3]}}%
3180   \def\@Acrshort##1##2[##3]{%

```

```

3181 \acronymfont{\@Glstext@{##1}{##2}[##3]}}%
3182 \def\@ACRshort##1##2[##3]{%
3183 \acronymfont{\@GLStext@{##1}{##2}[##3]}}%

Plural short form:
3184 \def\@acrshortpl##1##2[##3]{%
3185 \acronymfont{\@glsplural@{##1}{##2}[##3]}}%
3186 \def\@Acrshortpl##1##2[##3]{%
3187 \acronymfont{\@Glsplural@{##1}{##2}[##3]}}%
3188 \def\@ACRshortpl##1##2[##3]{%
3189 \acronymfont{\@GLSplural@{##1}{##2}[##3]}}%

Long form:
3190 \def\@acrlong##1##2[##3]{\@glsfirst@{##1}{##2}[##3]}}%
3191 \def\@Acrlong##1##2[##3]{\@Glsfirst@{##1}{##2}[##3]}}%
3192 \def\@ACRlong##1##2[##3]{\@GLSfirst@{##1}{##2}[##3]}}%

Plural long form:
3193 \def\@acrlongpl##1##2[##3]{\@glsfirstplural@{##1}{##2}[##3]}}%
3194 \def\@Acrlongpl##1##2[##3]{\@Glsfirstplural@{##1}{##2}[##3]}}%
3195 \def\@ACRlongpl##1##2[##3]{\@GLSfirstplural@{##1}{##2}[##3]}}%

Full form:
3196 \def\@acrfull##1##2[##3]{\@glsfirst@{##1}{##2}[##3]
3197 (\acronymfont{\@Glstext@{##1}{##2}[##3]}})%
3198 \def\@Acrfull##1##2[##3]{\@Glsfirst@{##1}{##2}[##3]
3199 (\acronymfont{\@Glstext@{##1}{##2}[##3]}})%
3200 \def\@ACRfull##1##2[##3]{\@GLSfirst@{##1}{##2}[##3]
3201 (\acronymfont{\@GLStext@{##1}{##2}[##3]}})%

Plural full form:
3202 \def\@acrfullpl##1##2[##3]{\@glsfirstplural@{##1}{##2}[##3]
3203 (\acronymfont{\@glsplural@{##1}{##2}[##3]}}}%
3204 \def\@Acrfullpl##1##2[##3]{\@Glsfirstplural@{##1}{##2}[##3]
3205 (\acronymfont{\@glsplural@{##1}{##2}[##3]}}}%
3206 \def\@ACRfullpl##1##2[##3]{\@GLSfirstplural@{##1}{##2}[##3]
3207 (\acronymfont{\@GLSplural@{##1}{##2}[##3]}}}%

check for option clash
3208 \ifglsacrdua
3209 \ifglsacrsmalldcaps
3210 \PackageError{glossaries}{Option clash: ‘smallcaps’ and ‘dua’
3211 can’t both be set}{}%
3212 \else
3213 \PackageError{glossaries}{Option clash: ‘smaller’ and ‘dua’
3214 can’t both be set}{}%
3215 \fi
3216 \fi
3217 }%

```

`\SetDUADisplayStyle` Sets the acronym display style for given glossary with dua setting.

```

3218 \newcommand*{\SetDUADisplayStyle}[1]{%

```

```

3219 \defglsdisplay[#1]{##1##4}%
3220 \defglsdisplayfirst[#1]{##1##4}%
3221 }

```

\DUANewAcronymDef

```

3222 \newcommand*{\DUANewAcronymDef}{%
3223   \edef\@do@newglossaryentry{%
3224     \noexpand\newglossaryentry{\the\glslabeltok}%
3225     {%
3226       type=\acronymtype,%
3227       name={\the\glsshorttok},%
3228       text={\the\glslongtok},%
3229       plural={\the\glslongtok\noexpand\acrpluralsuffix},%
3230       description={\the\glslongtok},%
3231       symbol={\the\glsshorttok},%
3232       symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%
3233       \the\glskeylisttok
3234     }%
3235   }%
3236   \@do@newglossaryentry
3237 }

```

\SetDUASStyle Always expand acronyms.

```

3238 \newcommand*{\SetDUASStyle}{%
3239   \renewcommand{\newacronym}[4][{}]{%
3240     \ifx\@glsacronymlists\@empty
3241       \def\@gls@type{\acronymtype}%
3242       \setkeys{glossentry}{##1}%
3243       \DeclareAcronymList{\@gls@type}%
3244       \SetDUADisplayStyle{\@gls@type}%
3245     \fi
3246     \glskeylisttok{##1}%
3247     \glslabeltok{##2}%
3248     \glsshorttok{##3}%
3249     \glslongtok{##4}%
3250     \newacronymhook
3251     \DUANewAcronymDef
3252   }%

```

Set up the commands to make a note of the keys to store the long and short forms:

```

3253 \def\glsshortkey{symbol}%
3254 \def\glsshortpluralkey{symbolplural}%
3255 \def\glslongkey{text}%
3256 \def\glslongpluralkey{plural}%

```

Set the display

```

3257 \@for\@gls@type:=\@glsacronymlists\do{%
3258   \SetDUADisplayStyle{\@gls@type}%
3259 }%

```

Set up short cuts. Short form:

```
3260 \def\@acrshort##1##2[##3]{\@glssymbol@{##1}-{##2}[##3]}%
3261 \def\@Acrshort##1##2[##3]{\@Glssymbol@{##1}-{##2}[##3]}%
3262 \def\@ACRshort##1##2[##3]{\@GLSsymbol@{##1}-{##2}[##3]}%
```

Plural short form:

```
3263 \def\@acrshortpl##1##2[##3]{\@glssymbolplural@{##1}-{##2}[##3]}%
3264 \def\@Acrshortpl##1##2[##3]{\@Glssymbolplural@{##1}-{##2}[##3]}%
3265 \def\@ACRshortpl##1##2[##3]{\@GLSsymbolplural@{##1}-{##2}[##3]}%
```

Long form:

```
3266 \def\@acrlong##1##2[##3]{\@glstext@{##1}-{##2}[##3]}%
3267 \def\@Acrlong##1##2[##3]{\@Glstext@{##1}-{##2}[##3]}%
3268 \def\@ACRlong##1##2[##3]{\@GLStext@{##1}-{##2}[##3]}%
```

Plural long form:

```
3269 \def\@acrlongpl##1##2[##3]{\@glsplural@{##1}-{##2}[##3]}%
3270 \def\@Acrlongpl##1##2[##3]{\@Glsplural@{##1}-{##2}[##3]}%
3271 \def\@ACRlongpl##1##2[##3]{\@GLSplural@{##1}-{##2}[##3]}%
```

Full form:

```
3272 \def\@acrfull##1##2[##3]{\@glstext@{##1}-{##2}[##3]
3273   (\acronymfont{\@glssymbol@{##1}-{##2}[##3]})}%
3274 \def\@Acrfull##1##2[##3]{\@Glstext@{##1}-{##2}[##3]
3275   (\acronymfont{\@glssymbol@{##1}-{##2}[##3]})}%
3276 \def\@ACRfull##1##2[##3]{\@GLStext@{##1}-{##2}[##3]
3277   (\acronymfont{\@GLSsymbol@{##1}-{##2}[##3]})}%
```

Plural full form:

```
3278 \def\@acrfullpl##1##2[##3]{\@glsplural@{##1}-{##2}[##3]
3279   (\acronymfont{\@glssymbolplural@{##1}-{##2}[##3]})}%
3280 \def\@Acrfullpl##1##2[##3]{\@Glsplural@{##1}-{##2}[##3]
3281   (\acronymfont{\@glssymbolplural@{##1}-{##2}[##3]})}%
3282 \def\@ACRfullpl##1##2[##3]{\@GLSplural@{##1}-{##2}[##3]
3283   (\acronymfont{\@GLSsymbolplural@{##1}-{##2}[##3]})}%
3284 }%
```

\SetAcronymStyle

```
3285 \newcommand*{\SetAcronymStyle}{%
3286   \SetDefaultAcronymStyle
3287   \ifglacrdescription
3288     \ifglacrfootnote
3289       \SetDescriptionFootnoteAcronymStyle
3290     \else
3291       \ifglacrdua
3292         \SetDescriptionDUAAcronymStyle
3293       \else
3294         \SetDescriptionAcronymStyle
3295       \fi
3296     \fi
3297   \else
```

```

3298 \ifglsacrfootnote
3299 \SetFootnoteAcronymStyle
3300 \else
3301 \ifthenelse{\boolean{glsacrsmallcaps}}\OR
3302 \boolean{glsacrsmaller}}%
3303 {%
3304 \SetSmallAcronymStyle
3305 }%
3306 {%
3307 \ifglsacrdua
3308 \SetDUASStyle
3309 \fi
3310 }%
3311 \fi
3312 \fi
3313 }

```

Set the acronym style according to the package options

```

3314 \SetAcronymStyle

```

Allow user to define their own custom acronyms. The short form is stored in the user1 key, the plural short form is stored in the user2 key, the long form is stored in the user3 key and the plural long form is stored in the user4 key. Defaults to displaying only the acronym with the long form as the description.

`\SetCustomDisplayStyle` Sets the acronym display style.

```

3315 \newcommand*{\SetCustomDisplayStyle}[1]{%
3316 \defglsdisplay[#1]{##1##4}%
3317 \defglsdisplayfirst[#1]{##1##4}%
3318 }

```

`\CustomAcronymFields`

```

3319 \newcommand*{\CustomAcronymFields}{%
3320 name={\the\glsshorttok},%
3321 description={\the\glslongtok},%
3322 first={\the\glslongtok\space(\the\glsshorttok)},%
3323 firstplural={\the\glslongtok\noexpand\acrpluralsuffix\space
3324 (\the\glsshorttok)}%
3325 text={\the\glsshorttok},%
3326 plural={\the\glsshorttok\noexpand\acrpluralsuffix}%
3327 }

```

`\CustomNewAcronymDef`

```

3328 \newcommand*{\CustomNewAcronymDef}{%
3329 \protected@edef\@do@newglossaryentry{%
3330 \noexpand\newglossaryentry{\the\glslabeltok}%
3331 {%
3332 type=\acronymtype,%
3333 user1={\the\glsshorttok},%
3334 user2={\the\glsshorttok\noexpand\acrpluralsuffix},%

```

```

3335     user3={\the\glslongtok},%
3336     user4={\the\glslongtok\noexpand\acrpluralsuffix},%
3337     \CustomAcronymFields,%
3338     \the\glskeylisttok
3339   }%
3340 }%
3341 \@do@newglossaryentry
3342 }

```

\SetCustomStyle

```

3343 \newcommand*{\SetCustomStyle}{%
3344   \renewcommand{\newacronym}[4][]{%
3345     \ifx\@glsacronymlists\@empty
3346       \def\@gls@type{\acronymtype}%
3347       \setkeys{glossentry}{##1}%
3348       \DeclareAcronymList{\@gls@type}%
3349       \SetCustomDisplayStyle{\@gls@type}%
3350     \fi
3351     \glskeylisttok{##1}%
3352     \glslabeltok{##2}%
3353     \glsshorttok{##3}%
3354     \glslongtok{##4}%
3355     \newacronymhook
3356     \CustomNewAcronymDef
3357   }%

```

Set up the commands to make a note of the keys to store the long and short forms:

```

3358   \def\glsshortkey{user1}%
3359   \def\glsshortpluralkey{user2}%
3360   \def\glslongkey{user3}%
3361   \def\glslongpluralkey{user4}%

```

Set the display

```

3362   \@for\@gls@type:=\@glsacronymlists\do{%
3363     \SetCustomDisplayStyle{\@gls@type}%
3364   }%

```

Set up short cuts. Short form:

```

3365   \def\@acrshort##1##2[##3]{\@glsuseri@{##1}{##2}[##3]}%
3366   \def\@Acrshort##1##2[##3]{\@Glsuseri@{##1}{##2}[##3]}%
3367   \def\@ACRshort##1##2[##3]{\@GLSuseri@{##1}{##2}[##3]}%

```

Plural short form:

```

3368   \def\@acrshortpl##1##2[##3]{\@glsuserii@{##1}{##2}[##3]}%
3369   \def\@Acrshortpl##1##2[##3]{\@Glsuserii@{##1}{##2}[##3]}%
3370   \def\@ACRshortpl##1##2[##3]{\@GLSuserii@{##1}{##2}[##3]}%

```

Long form:

```

3371   \def\@acrlong##1##2[##3]{\@glsuseriii@{##1}{##2}[##3]}%
3372   \def\@Acrlong##1##2[##3]{\@Glsuseriii@{##1}{##2}[##3]}%
3373   \def\@ACRlong##1##2[##3]{\@GLSuseriii@{##1}{##2}[##3]}%

```

Plural long form:

```
3374 \def\@acrlongpl##1##2[##3]{\@glsuseriv@{##1}-{##2}[##3]}%
3375 \def\@Acrlongpl##1##2[##3]{\@Glsuseriv@{##1}-{##2}[##3]}%
3376 \def\@ACRlongpl##1##2[##3]{\@GLSuseriv@{##1}-{##2}[##3]}%
```

Full form:

```
3377 \def\@acrfull##1##2[##3]{\@glsuseriii@{##1}-{##2}[##3]
3378 (\acronymfont{\@glsuseri@{##1}-{##2}[##3]})}%
3379 \def\@Acrfull##1##2[##3]{\@Glsuseriii@{##1}-{##2}[##3]
3380 (\acronymfont{\@glsuseri@{##1}-{##2}[##3]})}%
3381 \def\@ACRfull##1##2[##3]{\@GLSuseriii@{##1}-{##2}[##3]
3382 (\acronymfont{\@GLSuseri@{##1}-{##2}[##3]})}%
```

Plural full form:

```
3383 \def\@acrfullpl##1##2[##3]{\@glsuseriv@{##1}-{##2}[##3]
3384 (\acronymfont{\@glsuserii@{##1}-{##2}[##3]})}%
3385 \def\@Acrfullpl##1##2[##3]{\@Glsuseriv@{##1}-{##2}[##3]
3386 (\acronymfont{\@glsuserii@{##1}-{##2}[##3]})}%
3387 \def\@ACRfullpl##1##2[##3]{\@GLSuseriv@{##1}-{##2}[##3]
3388 (\acronymfont{\@GLSuserii@{##1}-{##2}[##3]})}%
3389 }%
```

\DefineAcronymSynonyms

```
3390 \newcommand*{\DefineAcronymSynonyms}{%
```

Short form

\acs

```
3391 \let\acs\acrshort
```

First letter uppercase short form

\Acs

```
3392 \let\Acs\Acrshort
```

Plural short form

\acsp

```
3393 \let\acsp\acrshortpl
```

First letter uppercase plural short form

\Acsp

```
3394 \let\Acsp\Acrshortpl
```

Long form

\acl

```
3395 \let\acl\aclong
```

Plural long form

`\aclp`
 3396 `\let\aclp\acrlongpl`
 First letter upper case long form

`\Acl`
 3397 `\let\Acl\Acrlong`
 First letter upper case plural long form

`\Aclp`
 3398 `\let\Aclp\Acrlongpl`
 Full form

`\acf`
 3399 `\let\acf\acrfull`
 Plural full form

`\acfp`
 3400 `\let\acfp\acrfullpl`
 First letter upper case full form

`\Acf`
 3401 `\let\Acf\Acrfull`
 First letter upper case plural full form

`\Acfp`
 3402 `\let\Acfp\Acrfullpl`
 Standard form

`\ac`
 3403 `\let\ac\gls`
 First upper case standard form

`\Ac`
 3404 `\let\Ac\Gls`
 Standard plural form

`\acp`
 3405 `\let\acp\glspl`
 Standard first letter upper case plural form

`\Acp`
 3406 `\let\Acp\Glspl`


```
3407 }
```

Define synonyms if required

```
3408 \ifglsacrshortcuts
```

```
3409   \DefineAcronymSynonyms
```

```
3410 \fi
```

4.18 Predefined Glossary Styles

The `glossaries` bundle comes with some predefined glossary styles. These need to be loaded now for the `style` option to use them.

First, the glossary hyper-navigation commands need to be loaded.

```
3411 \RequirePackage{glossary-hypernav}
```

The styles that use list-like environments. These are not loaded if the `nolist` option is used:

```
3412 \@gls@loadlist
```

The styles that use the `longtable` environment. These are not loaded if the `nolong` package option is used.

```
3413 \@gls@loadlong
```

The styles that use the `supertabular` environment. These are not loaded if the `nosuper` package option is used or if the package isn't installed.

```
3414 \@gls@loadsuper
```

The tree-like styles. These are not loaded if the `notree` package option is used.

```
3415 \@gls@loadtree
```

The default glossary style is set according to the `style` package option, but can be overridden by `\glossarystyle`. The required style must be defined at this point.

```
3416 \ifx\@glossary@default@style\relax
```

```
3417 \else
```

```
3418   \glossarystyle{\@glossary@default@style}
```

```
3419 \fi
```

5 Mfirstuc Documented Code

```
3420 \NeedsTeXFormat{LaTeX2e}
```

```
3421 \ProvidesPackage{mfirstuc}[2009/11/03 v1.04 (NLCT)]
```

`\makefirstuc` Syntax:

```
\makefirstuc{<text>}
```

Makes the first letter uppercase, but will skip initial control sequences if they are followed by a group and make the first thing in the group uppercase, unless the group is empty. Thus `\makefirstuc{abc}` will produce: `Abc`, `\makefirstuc{\ae bc}` will produce: `Æbc`, but `\makefirstuc{\emph{abc}}` will produce `Abc`. This is required by `\Gls` and `\Glspl`.

```
3422 \newif\if@glscs
```

```
3423 \newtoks\@glsmfirst
```

```

3424 \newtoks\@glsmrest
3425 \def\makefirstuc#1{%
3426   \def\gls@argi{#1}%
3427   \ifx\gls@argi\@empty
      If the argument is empty, do nothing.
3428   \else

3429     \def\@gls@tmp{\ #1}%
3430     \@onelevel@sanitize\@gls@tmp
3431     \expandafter\@gls@checkcs\@gls@tmp\relax\relax
3432     \if@glscs
3433       \@gls@getbody #1}\@nil
3434       \ifx\@gls@rest\@empty
3435         \@gls@makefirstuc{#1}%
3436       \else
3437         \expandafter\@gls@split\@gls@rest\@nil
3438         \ifx\@gls@first\@empty
3439           \@gls@makefirstuc{#1}%
3440         \else
3441           \expandafter\@glsmfirst\expandafter{\@gls@first}%
3442           \expandafter\@glsmrest\expandafter{\@gls@rest}%
3443           \edef\@gls@domfirstuc{\noexpand\@gls@body
3444             {\noexpand\@gls@makefirstuc\the\@glsmfirst}%
3445             \the\@glsmrest}%
3446           \@gls@domfirstuc
3447         \fi
3448       \fi
3449     \else
3450       \@gls@makefirstuc{#1}%
3451     \fi
3452 \fi
3453 }

      Put first argument in \@gls@first and second argument in \@gls@rest:
3454 \def\@gls@split#1#2\@nil{%
3455   \def\@gls@first{#1}\def\@gls@rest{#2}%
3456 }

3457 \def\@gls@checkcs#1 #2#3\relax{%
3458   \def\@gls@argi{#1}\def\@gls@argii{#2}%
3459   \ifx\@gls@argi\@gls@argii
3460     \@glscstrue
3461   \else
3462     \@glscsfalse
3463   \fi
3464 }

      Make first thing upper case:
3465 \def\@gls@makefirstuc#1{\MakeUppercase #1}

```

Get the first grouped argument and stores in `\@gls@body`.

```

3466 \def\@gls@getbody#1#\def\@gls@body{#1}\@gls@gobbletonil}
    Scoup up everything to \@nil and store in \@gls@rest:
3467 \def\@gls@gobbletonil#1\@nil{\def\@gls@rest{#1}}

\makefirstuc Expand argument once before applying \makefirstuc (added v1.01).
3468 \newcommand*\@makefirstuc[1]{%
3469 \expandafter\makefirstuc\expandafter{#1}}

```

6 Glossary Styles

6.1 Glossary hyper-navigation definitions (glossary-hypernav package)

Package Definition:

```

3470 \ProvidesPackage{glossary-hypernav}[2007/07/04 v1.01 (NLCT)]

```

The commands defined in this package are provided to help navigate around the groups within a glossary (see [subsection 4.15](#).) `\printglossary` (and `\printglossaries`) set `\@glo@type` to the label of the current glossary. This is used to create a unique hypertarget in the event of multiple glossaries.

```
\glsnavhyperlink[⟨type⟩]{⟨label⟩}{⟨text⟩}
```

This command makes `⟨text⟩` a hyperlink to the glossary group whose label is given by `⟨label⟩` for the glossary given by `⟨type⟩`.

```
\glsnavhyperlink
```

```

3471 \newcommand*\glsnavhyperlink[3][\@glo@type]{%
3472 \edef\gls@grplabel{#2}\protected@edef\gls@grptitle{#3}%
3473 \@glslink{glsn:#1@#2}{#3}}

```

```
\glsnavhypertarget[⟨type⟩]{⟨label⟩}{⟨text⟩}
```

This command makes `⟨text⟩` a hypertarget for the glossary group whose label is given by `⟨label⟩` in the glossary given by `⟨type⟩`. If `⟨type⟩` is omitted, `\@glo@type` is used which is set by `\printglossary` to the current glossary label.

```
\glsnavhypertarget
```

```

3474 \newcommand*\glsnavhypertarget[3][\@glo@type]{%
    Add this group to the aux file for re-run check.
3475 \protected@write\@auxout{}\string\gls@hypergroup{#1}{#2}}%
    Add the target.
3476 \@glstarget{glsn:#1@#2}{#3}%

```

Check list of know groups to determine if a re-run is required.

```

3477 \expandafter\let
3478 \expandafter\@gls@list\csname @gls@hypergroup\list@#1\endcsname
    Iterate through list and terminate loop if this group is found.
3479 \@for\@gls@elem:=\@gls@list\do{%
3480 \ifthenelse{\equal{\@gls@elem}{#2}}{\@endfortrue}{}}%
    Check if list terminated prematurely.
3481 \if@endfor
3482 \else
    This group was not included in the list, so issue a warning.
3483 \GlossariesWarningNoLine{Navigation panel
3484 for glossary type '#1'~Jmissing group '#2'}%
3485 \gdef\gls@hypergroup\prerun{%
3486 \GlossariesWarningNoLine{Navigation panel
3487 has changed. Rerun LaTeX}}%
3488 \fi
3489 }

```

`\gls@hypergroup\prerun` Give a warning at the end if re-run required

```

3490 \let\gls@hypergroup\prerun\relax
3491 \AtEndDocument{\gls@hypergroup\prerun}

```

`\@gls@hypergroup` This adds to (or creates) the command `\@gls@hypergroup\list@{glossary type}` which lists all groups for a given glossary, so that the navigation bar only contains those groups that are present. However it requires at least 2 runs to ensure the information is up-to-date.

```

3492 \newcommand*{\@gls@hypergroup}[2]{%
3493 \@ifundefined{\@gls@hypergroup\list@#1}{%
3494 \expandafter\xdef\csname @gls@hypergroup\list@#1\endcsname{#2}%
3495 }{%
3496 \expandafter\let\expandafter\@gls@tmp
3497 \csname @gls@hypergroup\list@#1\endcsname
3498 \expandafter\xdef\csname @gls@hypergroup\list@#1\endcsname{%
3499 \@gls@tmp,#2}%
3500 }%
3501 }

```

The `\glsnavigation` command displays a simple glossary group navigation. The symbol and number elements are defined separately, so that they can be suppressed if need be. Note that this command will produce a link to all 28 groups, but some groups may not be defined if there are groups that do not contain any terms, in which case you will get an undefined hyperlink warning. Now for the whole navigation bit:

`\glsnavigation`

```

3502 \newcommand*{\glsnavigation}{%
3503 \def\@gls@between{}%

```

```

3504 \@ifundefined{gls@hypergrouplist@%glo@type}{%
3505     \def@gls@list{}%
3506 }{%
3507     \expandafter\let\expandafter@gls@list
3508         \csname gls@hypergrouplist@%glo@type\endcsname
3509 }%
3510 \@for@gls@tmp:=@gls@list\do{%
3511     @gls@between
3512     \glsnavhyperlink{@gls@tmp}{\glsgetgrouptitle{@gls@tmp}}%
3513     \let@gls@between\glshypernavsep%
3514 }%
3515 }

```

`\glshypernavsep` Separator for the hyper navigation bar.

```

3516 \newcommand*{\glshypernavsep}{\space\textbar\space}

```

The `\glssymbolnav` produces a simple navigation set of links for just the symbol and number groups. This used to be used at the start of `\glsnavigation`. This command is no longer needed.

`\glssymbolnav`

```

3517 \newcommand*{\glssymbolnav}{%
3518     \glsnavhyperlink{glsymbols}{\glsgetgrouptitle{glsymbols}}%
3519     \glshypernavsep
3520     \glsnavhyperlink{glsnumbers}{\glsgetgrouptitle{glsnumbers}}%
3521     \glshypernavsep
3522 }

```

6.2 List Style (glossary-list.sty)

The style file defines glossary styles that use the `description` environment. Note that since the entry name is placed in the optional argument to the `\item` command, it will appear in a bold font by default.

```

3523 \ProvidesPackage{glossary-list}[2009/05/30 v2.01 (NLCT)]

```

- list** The list glossary style uses the `description` environment. The group separator `\glsgroupskip` is redefined as `\indexspace` which produces a gap between groups. The glossary heading and the group headings do nothing. Sub-entries immediately follow the main entry without the sub-entry name. This style does not use the entry's symbol. This is used as the default style for the `glossaries` package.

```

3524 \newglossarystyle{list}{%
    Use description environment:
3525     \renewenvironment{theglossary}%
3526         {\begin{description}}{\end{description}}%
    No header at the start of the environment:
3527     \renewcommand*{\glossaryheader}{}%

```

No group headings:

```
3528 \renewcommand*{\glsgroupeheading}[1]{}%
Main (level 0) entries start a new item in the list:
3529 \renewcommand*{\glossaryentryfield}[5]{%
3530 \item[\glstarget{##1}{##2}] ##3\glspostdescription\space ##5}%
Sub-entries continue on the same line:
3531 \renewcommand*{\glossarysubentryfield}[6]{%
3532 \glstarget{##2}{\strut}##4\glspostdescription\space ##6.}%
3533 % \end{macrocode}
3534 % Add vertical space between groups:
3535 % \begin{macrocode}
3536 \renewcommand*{\glsgroupskip}{\indexspace}%
3537 }
```

listgroup The listgroup style is like the list style, but the glossary groups have headings.

```
3538 \newglossarystyle{listgroup}{%
Base it on the list style:
3539 \glossarystyle{list}%
Each group has a heading:
3540 \renewcommand*{\glsgroupeheading}[1]{\item[\glsetgrouptitle{##1}]}}
```

listhypergroup The listhypergroup style is like the listgroup style, but has a set of links to the groups at the start of the glossary.

```
3541 \newglossarystyle{listhypergroup}{%
Base it on the list style:
3542 \glossarystyle{list}%
Add navigation links at the start of the environment:
3543 \renewcommand*{\glossaryheader}{%
3544 \item[\glsnavigation]}%
Each group has a heading with a hypertarget:
3545 \renewcommand*{\glsgroupeheading}[1]{%
3546 \item[\glsnavhypertarget{##1}{\glsetgrouptitle{##1}]}}}
```

altlist The altlist glossary style is like the list style, but places the description on a new line. Sub-entries follow in separate paragraphs without the sub-entry name. This style does not use the entry's symbol.

```
3547 \newglossarystyle{altlist}{%
Base it on the list style:
3548 \glossarystyle{list}%
Main (level 0) entries start a new item in the list with a line break after the entry name:
3549 \renewcommand*{\glossaryentryfield}[5]{%
3550 \item[\glstarget{##1}{##2}]\mbox{}\\
3551 ##3\glspostdescription\space ##5}%

```

Sub-entries start a new paragraph:

```
3552 \renewcommand{\glossarysubentryfield}[6]{%
3553   \par\glstarget{##2}{\strut}##4\glspostdescription\space ##6}%
3554 }
```

altlistgroup The altlistgroup glossary style is like the altlist style, but the glossary groups have headings.

```
3555 \newglossarystyle{altlistgroup}{%
```

Base it on the altlist style:

```
3556 \glossarystyle{altlist}%
```

Each group has a heading:

```
3557 \renewcommand*{\glsgroupheading}[1]{\item[\glsgrouptitle{##1}]}
```

altlisthypergroup The altlisthypergroup glossary style is like the altlistgroup style, but has a set of links to the groups at the start of the glossary.

```
3558 \newglossarystyle{altlisthypergroup}{%
```

Base it on the altlist style:

```
3559 \glossarystyle{altlist}%
```

Add navigation links at the start of the environment:

```
3560 \renewcommand*{\glossaryheader}{%
```

```
3561   \item[\glsnavigation]}%
```

Each group has a heading with a hypertext:

```
3562 \renewcommand*{\glsgroupheading}[1]{%
```

```
3563   \item[\glsnavhypertarget{##1}{\glsgrouptitle{##1}}]}
```

listdotted The listdotted glossary style was supplied by Axel Menzel. I've modified it slightly so that the distance from the start of the name to the end of the dotted line is specified by `\glslistdottedwidth`. Note that this style ignores the page numbers as well as the symbol. Sub-entries are displayed in the same way as top-level entries.

```
3564 \newglossarystyle{listdotted}{%
```

Base it on the list style:

```
3565 \glossarystyle{list}%
```

Each main (level 0) entry starts a new item:

```
3566 \renewcommand*{\glossaryentryfield}[5]{%
```

```
3567   \item[\makebox[\glslistdottedwidth][l]{\glstarget{##1}{##2}}%
```

```
3568   \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##3}%
```

Sub entries have the same format as main entries:

```
3569 \renewcommand*{\glossarysubentryfield}[6]{%
```

```
3570   \item[\makebox[\glslistdottedwidth][l]{\glstarget{##2}{##3}}%
```

```
3571   \unskip\leaders\hbox to 2.9mm{\hss.}\hfill\strut}##4}%
```

```
3572 }
```

`\glslistdottedwidth`

```
3573 \newlength\glslistdottedwidth
3574 \setlength{\glslistdottedwidth}{.5\hsize}
```

`sublistdotted` This style is similar to the `glostylelistdotted` style, except that the main entries just have the name displayed.

```
3575 \newglossarystyle{sublistdotted}{%
```

Base it on the `listdotted` style:

```
3576 \glossarystyle{listdotted}%
```

Main (level 0) entries just display the name:

```
3577 \renewcommand*{\glossaryentryfield}[5]{%
3578   \item[\glstarget{##1}{##2}]}%
3579 }
```

6.3 Glossary Styles using `longtable` (the `glossary-long` package)

The glossary styles defined in the package used the `longtable` environment in the glossary.

```
3580 \ProvidesPackage{glossary-long}[2009/05/30 v2.01 (NLCT)]
```

Requires the package:

```
3581 \RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. (There's a chance that the user may specify `nolong` and then load later, in which case `\glsdescwidth` may have already been defined by . The same goes for `\glspagelistwidth`.)

```
3582 \@ifundefined{glsdescwidth}{%
3583   \newlength\glsdescwidth
3584   \setlength{\glsdescwidth}{0.6\hsize}
3585 }
```

`\glspagelistwidth` This is a length that governs the width of the page list column.

```
3586 \@ifundefined{glspagelistwidth}{%
3587   \newlength\glspagelistwidth
3588   \setlength{\glspagelistwidth}{0.1\hsize}
3589 }
```

`long` The long glossary style command which uses the `longtable` environment:

```
3590 \newglossarystyle{long}{%
```

Use `longtable` with two columns:

```
3591 \renewenvironment{theglossary}%
3592   {\begin{longtable}[lp{\glsdescwidth}]}%
3593   {\end{longtable}}%
```


Do nothing at the start of the environment:

```
3594 \renewcommand*{\glossaryheader}{}%
```

No heading between groups:

```
3595 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries displayed in a row:

```
3596 \renewcommand*{\glossaryentryfield}[5]{%
```

```
3597   \glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
```

Sub entries displayed on the following row without the name:

```
3598 \renewcommand*{\glossarysubentryfield}[6]{%
```

```
3599   & \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
```

Blank row between groups:

```
3600 \renewcommand*{\glsgroupskip}{ & \\}%
```

```
3601 }
```

longborder The **longborder** style is like the above, but with horizontal and vertical lines:

```
3602 \newglossarystyle{longborder}{%
```

Base it on the **glostylelong** style:

```
3603 \glossarystyle{long}%
```

Use **longtable** with two columns with vertical lines between each column:

```
3604 \renewenvironment{theglossary}{%
```

```
3605   \begin{longtable}{|l|p{\glsdescwidth}|}{\end{longtable}}%
```

Place horizontal lines at the head and foot of the table:

```
3606 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
```

```
3607 }
```

longheader The **longheader** style is like the **long** style but with a header:

```
3608 \newglossarystyle{longheader}{%
```

Base it on the **glostylelong** style:

```
3609 \glossarystyle{long}%
```

Set the table's header:

```
3610 \renewcommand*{\glossaryheader}{%
```

```
3611   \bfseries \entryname & \bfseries \descriptionname\\endhead}%
```

```
3612 }
```

longheaderborder The **longheaderborder** style is like the **long** style but with a header and border:

```
3613 \newglossarystyle{longheaderborder}{%
```

Base it on the **glostylelongborder** style:

```
3614 \glossarystyle{longborder}%
```

Set the table's header and add horizontal line to table's foot:

```
3615 \renewcommand*{\glossaryheader}{%
```

```
3616   \hline\bfseries \entryname & \bfseries \descriptionname\\hline
```

```
3617   \endhead
```

```
3618   \hline\endfoot}%
```

```
3619 }
```

long3col The long3col style is like long but with 3 columns

```

3620 \newglossarystyle{long3col}{%
    Use a longtable with 3 columns:
3621 \renewenvironment{theglossary}%
3622     {\begin{longtable}{lp{\glsdescwidth}p{\glspagelistwidth}}}%
3623     {\end{longtable}}}%
    No table header:
3624 \renewcommand*{\glossaryheader}{}%
    No headings between groups:
3625 \renewcommand*{\glsgroupheading}[1]{}%
    Main (level 0) entries on a row (name in first column, description in second column,
    page list in last column):
3626 \renewcommand*{\glossaryentryfield}[5]{%
3627     \glstarget{##1}{##2} & ##3 & ##5\\}%
    Sub-entries on a separate row (no name, description in second column, page list
    in third column):
3628 \renewcommand*{\glossarysubentryfield}[6]{%
3629     & \glstarget{##2}{\strut}##4 & ##6\\}%
    Blank row between groups:
3630 \renewcommand*{\glsgroupskip}{ & &\\}%
3631 }

```

long3colborder The long3colborder style is like the long3col style but with a border:

```

3632 \newglossarystyle{long3colborder}{%
    Base it on the glostylelong3col style:
3633 \glossarystyle{long3col}%
    Use a longtable with 3 columns with vertical lines around them:
3634 \renewenvironment{theglossary}%
3635     {\begin{longtable}{|lp{\glsdescwidth}|p{\glspagelistwidth}|}}%
3636     {\end{longtable}}}%
    Place horizontal lines at the head and foot of the table:
3637 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
3638 }

```

long3colheader The long3colheader style is like long3col but with a header row:

```

3639 \newglossarystyle{long3colheader}{%
    Base it on the glostylelong3col style:
3640 \glossarystyle{long3col}%
    Set the table's header:
3641 \renewcommand*{\glossaryheader}{%
3642     \bfseries\entryname&\bfseries\descriptionname&
3643     \bfseries\pagelistname\\endhead}%
3644 }

```

long3colheaderborder The long3colheaderborder style is like the above but with a border

```
3645 \newglossarystyle{long3colheaderborder}{%
```

Base it on the glostylelong3colborder style:

```
3646 \glossarystyle{long3colborder}%
```

Set the table's header and add horizontal line at table's foot:

```
3647 \renewcommand*{\glossaryheader}{%
```

```
\hline
```

```
3649 \bfseries\entryname&\bfseries\descriptionname&
```

```
3650 \bfseries\pagelistname\\hline\endhead
```

```
3651 \hline\endfoot}%
```

```
3652 }
```

long4col The long4col style has four columns where the third column contains the value of the associated symbol key.

```
3653 \newglossarystyle{long4col}{%
```

Use a longtable with 4 columns:

```
3654 \renewenvironment{theglossary}%
```

```
3655 {\begin{longtable}{llll}}%
```

```
3656 {\end{longtable}}%
```

No table header:

```
3657 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
3658 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a single row (name in first column, description in second column, symbol in third column, page list in last column):

```
3659 \renewcommand*{\glossaryentryfield}[5]{%
```

```
3660 \glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
3661 \renewcommand*{\glossarysubentryfield}[6]{%
```

```
3662 & \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%
```

Blank row between groups:

```
3663 \renewcommand*{\glsgroupskip}{ & & &\\}%
```

```
3664 }
```

long4colheader The long4colheader style is like long4col but with a header row.

```
3665 \newglossarystyle{long4colheader}{%
```

Base it on the glostylelong4col style:

```
3666 \glossarystyle{long4col}%
```

Table has a header:

```
3667 \renewcommand*{\glossaryheader}{%
```

```
3668 \bfseries\entryname&\bfseries\descriptionname&
```

```
3669 \bfseries \symbolname&
```

```

3670 \bfseries\pagelistname\\\endhead}%
3671 }

```

long4colborder The long4colborder style is like long4col but with a border.

```

3672 \newglossarystyle{long4colborder}{%
    Base it on the glostylelong4col style:
3673 \glossarystyle{long4col}%
    Use a longtable with 4 columns surrounded by vertical lines:
3674 \renewenvironment{theglossary}%
3675 {\begin{longtable}{|l|l|l|l|}}%
3676 {\end{longtable}}%
    Add horizontal lines to the head and foot of the table:
3677 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
3678 }

```

long4colheaderborder The long4colheaderborder style is like the above but with a border.

```

3679 \newglossarystyle{long4colheaderborder}{%
    Base it on the glostylelong4col style:
3680 \glossarystyle{long4col}%
    Use a longtable with 4 columns surrounded by vertical lines:
3681 \renewenvironment{theglossary}%
3682 {\begin{longtable}{|l|l|l|l|}}%
3683 {\end{longtable}}%
    Add table header and horizontal line at the table's foot:
3684 \renewcommand*{\glossaryheader}{%
3685 \hline\bfseries\entryname&\bfseries\descriptionname&
3686 \bfseries \symbolname&
3687 \bfseries\pagelistname\\\hline\endhead\hline\endfoot}%
3688 }

```

altlong4col The altlong4col style is like the long4col style but can have multiline descriptions and page lists.

```

3689 \newglossarystyle{altlong4col}{%
    Base it on the glostylelong4col style:
3690 \glossarystyle{long4col}%
    Use a longtable with 4 columns where the second and last columns may have
    multiple lines in each row:
3691 \renewenvironment{theglossary}%
3692 {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%
3693 {\end{longtable}}%
3694 }

```

altlong4colheader The altlong4colheader style is like altlong4col but with a header row.

```

3695 \newglossarystyle{altlong4colheader}{%

```

Base it on the `glostylelong4colheader` style:

```
3696 \glossarystyle{long4colheader}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
3697 \renewenvironment{theglossary}%  
3698   {\begin{longtable}{lp{\glstdescwidth}lp{\glspagelistwidth}}}%  
3699   {\end{longtable}}}%  
3700 }
```

`altlong4colborder` The `altlong4colborder` style is like `altlong4col` but with a border.

```
3701 \newglossarystyle{altlong4colborder}{%
```

Base it on the `glostylelong4colborder` style:

```
3702 \glossarystyle{long4colborder}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
3703 \renewenvironment{theglossary}%  
3704   {\begin{longtable}{|lp{\glstdescwidth}|lp{\glspagelistwidth}|}}}%  
3705   {\end{longtable}}}%  
3706 }
```

`altlong4colheaderborder` The `altlong4colheaderborder` style is like the above but with a header as well as a border.

```
3707 \newglossarystyle{altlong4colheaderborder}{%
```

Base it on the `glostylelong4colheaderborder` style:

```
3708 \glossarystyle{long4colheaderborder}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
3709 \renewenvironment{theglossary}%  
3710   {\begin{longtable}{|lp{\glstdescwidth}|lp{\glspagelistwidth}|}}}%  
3711   {\end{longtable}}}%  
3712 }
```

6.4 Glossary Styles using `longtable` (the `glossary-longragged` package)

The glossary styles defined in the package used the `longtable` environment in the glossary and use ragged right formatting for the multiline columns.

```
3713 \ProvidesPackage{glossary-longragged}[2009/05/30 v2.01 (NLCT)]
```

Requires the package:

```
3714 \RequirePackage{array}
```

Requires the package:

```
3715 \RequirePackage{longtable}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may have already been defined.

```

3716 \ifundefined{glsdescwidth}{%
3717   \newlength{glsdescwidth
3718   \setlength{glsdescwidth}{0.6\hsize}
3719 }{}

```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined.

```

3720 \ifundefined{glspagelistwidth}{%
3721   \newlength{glspagelistwidth
3722   \setlength{glspagelistwidth}{0.1\hsize}
3723 }{}

```

`longragged` The `longragged` glossary style is like the `long` but uses ragged right formatting for the description column.

```

3724 \newglossarystyle{longragged}{%
    Use longtable with two columns:
3725   \renewenvironment{theglossary}{%
3726     {\begin{longtable}{l>{\raggedright}p{glsdescwidth}}}%
3727     {\end{longtable}}}%
    Do nothing at the start of the environment:
3728   \renewcommand*{\glossaryheader}{}%
    No heading between groups:
3729   \renewcommand*{\glsgroupheading}[1]{}%
    Main (level 0) entries displayed in a row:
3730   \renewcommand*{\glossaryentryfield}[5]{%
3731     \glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
3732     \tabularnewline}%
    Sub entries displayed on the following row without the name:
3733   \renewcommand*{\glossarysubentryfield}[6]{%
3734     & \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
3735     \tabularnewline}%
    Blank row between groups:
3736   \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
3737 }

```

`longraggedborder` The `longraggedborder` style is like the above, but with horizontal and vertical lines:

```

3738 \newglossarystyle{longraggedborder}{%
    Base it on the glostylelongragged style:
3739   \glossarystyle{longragged}%
    Use longtable with two columns with vertical lines between each column:
3740   \renewenvironment{theglossary}{%
3741     \begin{longtable}{|l|>{\raggedright}p{glsdescwidth}|}%
3742     {\end{longtable}}%

```

Place horizontal lines at the head and foot of the table:

```
3743 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
3744 }
```

longraggedheader The longraggedheader style is like the longragged style but with a header:

```
3745 \newglossarystyle{longraggedheader}{%
    Base it on the glostylelongragged style:
3746 \glossarystyle{longragged}%
    Set the table's header:
3747 \renewcommand*{\glossaryheader}{%
3748     \bfseries \entryname & \bfseries \descriptionname
3749     \tabularnewline\endhead}%
3750 }
```

longraggedheaderborder The longraggedheaderborder style is like the longragged style but with a header and border:

```
3751 \newglossarystyle{longraggedheaderborder}{%
    Base it on the glostylelongraggedborder style:
3752 \glossarystyle{longraggedborder}%
    Set the table's header and add horizontal line to table's foot:
3753 \renewcommand*{\glossaryheader}{%
3754     \hline\bfseries \entryname & \bfseries \descriptionname
3755     \tabularnewline\hline
3756     \endhead
3757     \hline\endfoot}%
3758 }
```

longragged3col The longragged3col style is like longragged but with 3 columns

```
3759 \newglossarystyle{longragged3col}{%
    Use a longtable with 3 columns:
3760 \renewenvironment{theglossary}%
3761     {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}%
3762      >{\raggedright}p{\glspagelistwidth}}}%
3763     {\end{longtable}}%
```

No table header:

```
3764 \renewcommand*{\glossaryheader}{}%
```

No headings between groups:

```
3765 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
3766 \renewcommand*{\glossaryentryfield}[5]{%
3767     \glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%

```

Sub-entries on a separate row (no name, description in second column, page list in third column):

```
3768 \renewcommand*{\glossarysubentryfield}[6]{%
3769     & \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
```

Blank row between groups:

```
3770 \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
3771 }
```

longragged3colborder The longragged3colborder style is like the longragged3col style but with a border:

```
3772 \newglossarystyle{longragged3colborder}{%
    Base it on the glostylelongragged3col style:
3773 \glossarystyle{longragged3col}%
    Use a longtable with 3 columns with vertical lines around them:
3774 \renewenvironment{theglossary}%
3775     {\begin{longtable}{|l|>{\raggedright}p{\glsgdescwidth}|%
3776      >{\raggedright}p{\glspagelistwidth}|}}%
3777     {\end{longtable}}}%
    Place horizontal lines at the head and foot of the table:
3778 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
3779 }
```

longragged3colheader The longragged3colheader style is like longragged3col but with a header row:

```
3780 \newglossarystyle{longragged3colheader}{%
    Base it on the glostylelongragged3col style:
3781 \glossarystyle{longragged3col}%
    Set the table's header:
3782 \renewcommand*{\glossaryheader}{%
3783     \bfseries\entryname&\bfseries\descriptionname&
3784     \bfseries\pagelistname\tabularnewline\endhead}%
3785 }
```

longragged3colheaderborder The longragged3colheaderborder style is like the above but with a border

```
3786 \newglossarystyle{longragged3colheaderborder}{%
    Base it on the glostylelongragged3colborder style:
3787 \glossarystyle{longragged3colborder}%
    Set the table's header and add horizontal line at table's foot:
3788 \renewcommand*{\glossaryheader}{%
3789     \hline
3790     \bfseries\entryname&\bfseries\descriptionname&
3791     \bfseries\pagelistname\tabularnewline\hline\endhead
3792     \hline\endfoot}%
3793 }
```


`altlongragged4col` The `altlongragged4col` style is like the `altlong4col` style defined in the package, except that ragged right formatting is used for the description and page list columns.

```
3794 \newglossarystyle{altlongragged4col}{%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
3795 \renewenvironment{theglossary}%
3796 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
3797 >{\raggedright}p{\glspagelistwidth}}}%
3798 {\end{longtable}}%
```

No table header:

```
3799 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
3800 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries on a single row (name in first column, description in second column, symbol in third column, page list in last column):

```
3801 \renewcommand*{\glossaryentryfield}[5]{%
3802 \glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%
```

Sub entries on a single row with no name (description in second column, symbol in third column, page list in last column):

```
3803 \renewcommand*{\glossarysubentryfield}[6]{%
3804 & \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%
```

Blank row between groups:

```
3805 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
3806 }
```

`altlongragged4colheader` The `altlongragged4colheader` style is like `altlongragged4col` but with a header row.

```
3807 \newglossarystyle{altlongragged4colheader}{%
```

Base it on the `glostylealtlongragged4col` style:

```
3808 \glossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
3809 \renewenvironment{theglossary}%
3810 {\begin{longtable}{l>{\raggedright}p{\glsdescwidth}l%
3811 >{\raggedright}p{\glspagelistwidth}}}%
3812 {\end{longtable}}%
```

Table has a header:

```
3813 \renewcommand*{\glossaryheader}{%
3814 \bfseries\entryname&\bfseries\descriptionname&
3815 \bfseries \symbolname&
3816 \bfseries\pagelistname\tabularnewline\endhead}%
3817 }
```

`altlongragged4colborder` The `altlongragged4colborder` style is like `altlongragged4col` but with a border.

```
3818 \newglossarystyle{altlongragged4colborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
3819 \glossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
3820 \renewenvironment{theglossary}%
3821   {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|l|}%
3822    >{\raggedright}p{\glspagelistwidth}|}%
3823   {\end{longtable}}%
```

Add horizontal lines to the head and foot of the table:

```
3824 \renewcommand*{\glossaryheader}{\hline\endhead\hline\endfoot}%
3825 }
```

`altlongragged4colheaderborder` The `altlongragged4colheaderborder` style is like the above but with a header as well as a border.

```
3826 \newglossarystyle{altlongragged4colheaderborder}{%
```

Base it on the `glostylealtlongragged4col` style:

```
3827 \glossarystyle{altlongragged4col}%
```

Use a `longtable` with 4 columns where the second and last columns may have multiple lines in each row:

```
3828 \renewenvironment{theglossary}%
3829   {\begin{longtable}{|l|>{\raggedright}p{\glsdescwidth}|l|}%
3830    >{\raggedright}p{\glspagelistwidth}|}%
3831   {\end{longtable}}%
```

Add table header and horizontal line at the table's foot:

```
3832 \renewcommand*{\glossaryheader}{%
3833   \hline\bfseries\entryname&\bfseries\descriptionname&
3834   \bfseries \symbolname&
3835   \bfseries\pagelistname\tabularnewline\hline\endhead
3836   \hline\endfoot}%
3837 }
```

6.5 Glossary Styles using supertabular environment (glossary-super package)

The glossary styles defined in the package use the `supertabular` environment.

```
3838 \ProvidesPackage{glossary-super}[2009/05/30 v2.01 (NLCT)]
```

Requires the package:

```
3839 \RequirePackage{supertabular}
```

`\glsdescwidth` This is a length that governs the width of the description column. This may already have been defined if has been loaded.

```
3840 \@ifundefined{glsdescwidth}{%
3841   \newlength{glsdescwidth
3842   \setlength{glsdescwidth}{0.6\hspace}
3843 }
```

`\glspagelistwidth` This is a length that governs the width of the page list column. This may already have been defined if has been loaded.

```
3844 \ifundefined{glspagelistwidth}{%
3845   \newlength{glspagelistwidth
3846   \setlength{glspagelistwidth}{0.1\hsize}
3847 }{}
```

super The super glossary style uses the `supertabular` environment (it uses lengths defined in the package.)

```
3848 \newglossarystyle{super}{%
```

Put the glossary in a `supertabular` environment with two columns and no head or tail:

```
3849   \renewenvironment{theglossary}%
3850   {\tablehead{}\tabletail{}}%
3851   \begin{supertabular}{lp{glstdescwidth}}%
3852   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
3853   \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
3854   \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries put in a row (name in first column, description and page list in second column):

```
3855   \renewcommand*{\glossaryentryfield}[5]{%
3856     \glstarget{##1}{##2} & ##3\glspostdescription\space ##5\\}%
```

Sub entries put in a row (no name, description and page list in second column):

```
3857   \renewcommand*{\glossarysubentryfield}[6]{%
3858     & \glstarget{##2}{\strut}##4\glspostdescription\space ##6\\}%
```

Blank row between groups:

```
3859   \renewcommand*{\glsgroupskip}{ & \\}%
3860 }
```

superborder The `superborder` style is like the above, but with horizontal and vertical lines:

```
3861 \newglossarystyle{superborder}{%
```

Base it on the `glostylesuper` style:

```
3862   \glossarystyle{super}%
```

Put the glossary in a `supertabular` environment with two columns and a horizontal line in the head and tail:

```
3863   \renewenvironment{theglossary}%
3864   {\tablehead{\hline}\tabletail{\hline}%
3865   \begin{supertabular}{|lp{glstdescwidth}|}%
3866   {\end{supertabular}}%
3867 }
```

superheader The superheader style is like the super style, but with a header:

```

3868 \newglossarystyle{superheader}{%
      Base it on the glostylesuper style:
3869 \glossarystyle{super}%
      Put the glossary in a supertabular environment with two columns, a header and
      no tail:
3870 \renewenvironment{theglossary}%
3871   {\tablehead{\bfseries \entryname & \bfseries \descriptionname\\}%
3872    \tabletail{}}%
3873   \begin{supertabular}{lp{\glsgdescwidth}}}%
3874   {\end{supertabular}}}%
3875 }
```

superheaderborder The superheaderborder style is like the super style but with a header and border:

```

3876 \newglossarystyle{superheaderborder}{%
      Base it on the glostylesuper style:
3877 \glossarystyle{super}%
      Put the glossary in a supertabular environment with two columns, a header and
      horizontal lines above and below the table:
3878 \renewenvironment{theglossary}%
3879   {\tablehead{\hline\bfseries \entryname &
3880               \bfseries \descriptionname\\\hline}%
3881    \tabletail{\hline}
3882    \begin{supertabular}{|lp{\glsgdescwidth}|}%
3883    {\end{supertabular}}}%
3884 }
```

super3col The super3col style is like the super style, but with 3 columns:

```

3885 \newglossarystyle{super3col}{%
      Put the glossary in a supertabular environment with three columns and no head
      or tail:
3886 \renewenvironment{theglossary}%
3887   {\tablehead{} \tabletail{}}%
3888   \begin{supertabular}{lp{\glsgdescwidth}p{\glspagelistwidth}}}%
3889   {\end{supertabular}}%
      Do nothing at the start of the table:
3890 \renewcommand*{\glossaryheader}{}%
      No group headings:
3891 \renewcommand*{\glsgroupheading}[1]{}%
      Main (level 0) entries on a row (name in first column, description in second column,
      page list in last column):
3892 \renewcommand*{\glossaryentryfield}[5]{%
3893   \glstarget{##1}{##2} & ##3 & ##5\\}%

```

Sub entries on a row (no name, description in second column, page list in last column):

```
3894 \renewcommand*{\glossarysubentryfield}[6]{%
3895     & \glstarget{##2}{\strut}##4 & ##6\\}%
```

Blank row between groups:

```
3896 \renewcommand*{\glsgroupskip}{ & &\\}%
3897 }
```

super3colborder The `super3colborder` style is like the `super3col` style, but with a border:

```
3898 \newglossarystyle{super3colborder}{%
    Base it on the glostylesuper3col style:
3899 \glossarystyle{super3col}%
    Put the glossary in a supertabular environment with three columns and a horizontal
    line in the head and tail:
3900 \renewenvironment{theglossary}%
3901     {\tablehead{\hline}\tabletail{\hline}%
3902     \begin{supertabular}{|l|p{\glsgdescwidth}|p{\glspagelistwidth}|}%
3903     {\end{supertabular}}}%
3904 }
```

super3colheader The `super3colheader` style is like the `super3col` style but with a header row:

```
3905 \newglossarystyle{super3colheader}{%
    Base it on the glostylesuper3col style:
3906 \glossarystyle{super3col}%
    Put the glossary in a supertabular environment with three columns, a header and
    no tail:
3907 \renewenvironment{theglossary}%
3908     {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
3909                 \bfseries\pagelistname\\}\tabletail{}}%
3910     \begin{supertabular}{\lp{\glsgdescwidth}p{\glspagelistwidth}}}%
3911     {\end{supertabular}}}%
3912 }
```

super3colheaderborder The `super3colheaderborder` style is like the `super3col` style but with a header and border:

```
3913 \newglossarystyle{super3colheaderborder}{%
    Base it on the glostylesuper3colborder style:
3914 \glossarystyle{super3colborder}%
    Put the glossary in a supertabular environment with three columns, a header with
    horizontal lines and a horizontal line in the tail:
3915 \renewenvironment{theglossary}%
3916     {\tablehead{\hline
3917                 \bfseries\entryname&\bfseries\descriptionname&
3918                 \bfseries\pagelistname\\ \hline}%
    }
```

```

3919     \tabletail{\hline}%
3920     \begin{supertabular}{|l|p{\glstdescwidth}|p{\glspagelistwidth}|}%
3921     {\end{supertabular}}%
3922 }

```

super4col The **super4col** glossary style has four columns, where the third column contains the value of the corresponding symbol key used when that entry was defined.

```

3923 \newglossarystyle{super4col}{%

```

Put the glossary in a **supertabular** environment with four columns and no head or tail:

```

3924 \renewenvironment{theglossary}%
3925 {\tablehead{}\tabletail{}}%
3926 \begin{supertabular}{|l|l|l|l|}%
3927 \end{supertabular}}%

```

Do nothing at the start of the table:

```

3928 \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

3929 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries on a row with the name in the first column, description in second column, symbol in third column and page list in last column:

```

3930 \renewcommand*{\glossaryentryfield}[5]{%
3931 \glstarget{##1}{##2} & ##3 & ##4 & ##5\\}%

```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```

3932 \renewcommand*{\glossarysubentryfield}[6]{%
3933 & \glstarget{##2}{\strut}##4 & ##5 & ##6\\}%

```

Blank row between groups:

```

3934 \renewcommand*{\glsgroupskip}{ & & \\}%
3935 }

```

super4colheader The **super4colheader** style is like the **super4col** but with a header row.

```

3936 \newglossarystyle{super4colheader}{%

```

Base it on the **glostylesuper4col** style:

```

3937 \glossarystyle{super4col}%

```

Put the glossary in a **supertabular** environment with four columns, a header and no tail:

```

3938 \renewenvironment{theglossary}%
3939 {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
3940 \bfseries\symbolname &
3941 \bfseries\pagelistname\\}%
3942 \tabletail{}}%
3943 \begin{supertabular}{|l|l|l|l|}%
3944 {\end{supertabular}}%
3945 }

```

super4colborder The **super4colborder** style is like the **super4col** but with a border.

```
3946 \newglossarystyle{super4colborder}{%
      Base it on the glostylesuper4col style:
3947 \glossarystyle{super4col}%
      Put the glossary in a supertabular environment with four columns and a horizontal
      line in the head and tail:
3948 \renewenvironment{theglossary}%
3949   {\tablehead{\hline}\tabletail{\hline}%
3950    \begin{supertabular}{|l|l|l|l|}%
3951    {\end{supertabular}}%
3952 }
```

super4colheaderborder The **super4colheaderborder** style is like the **super4col** but with a header and border.

```
3953 \newglossarystyle{super4colheaderborder}{%
      Base it on the glostylesuper4col style:
3954 \glossarystyle{super4col}%
      Put the glossary in a supertabular environment with four columns and a header
      bordered by horizontal lines and a horizontal line in the tail:
3955 \renewenvironment{theglossary}%
3956   {\tablehead{\hline\bfseries\entryname&\bfseries\descriptionname&
3957               \bfseries\symbolname &
3958               \bfseries\pagelistname\\ \hline}\tabletail{\hline}%
3959   \begin{supertabular}{|l|l|l|l|}%
3960   {\end{supertabular}}%
3961 }
```

altsuper4col The **altsuper4col** glossary style is like **super4col** but has provision for multiline descriptions.

```
3962 \newglossarystyle{altsuper4col}{%
      Base it on the glostylesuper4col style:
3963 \glossarystyle{super4col}%
      Put the glossary in a supertabular environment with four columns and no head or
      tail:
3964 \renewenvironment{theglossary}%
3965   {\tablehead{}\tabletail{}%
3966    \begin{supertabular}{lp{\glsdescwidth}lp{\glspagelistwidth}}%
3967   {\end{supertabular}}%
3968 }
```

altsuper4colheader The **altsuper4colheader** style is like the **altsuper4col** but with a header row.

```
3969 \newglossarystyle{altsuper4colheader}{%
      Base it on the glostylesuper4colheader style:
3970 \glossarystyle{super4colheader}%

```

Put the glossary in a `supertabular` environment with four columns, a header and no tail:

```

3971 \renewenvironment{theglossary}%
3972   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
3973     \bfseries\symbolname &
3974     \bfseries\pagelistname\\}\tabletail{}}%
3975   \begin{supertabular}{lp{\glsgdescwidth}lp{\glspagelistwidth}}}%
3976   {\end{supertabular}}}%
3977 }

```

altsuper4colborder The `altsuper4colborder` style is like the `altsuper4col` but with a border.

```

3978 \newglossarystyle{altsuper4colborder}{%
    Base it on the glostylesuper4colborder style:
3979 \glossarystyle{super4colborder}%
    Put the glossary in a supertabular environment with four columns and a horizontal
    line in the head and tail:
3980 \renewenvironment{theglossary}%
3981   {\tablehead{\hline}\tabletail{\hline}%
3982   \begin{supertabular}%
3983     {\lllp{\glsgdescwidth}\lllp{\glspagelistwidth}}}%
3984   {\end{supertabular}}}%
3985 }

```

altsuper4colheaderborder The `altsuper4colheaderborder` style is like the `altsuper4col` but with a header and border.

```

3986 \newglossarystyle{altsuper4colheaderborder}{%
    Base it on the glostylesuper4colheaderborder style:
3987 \glossarystyle{super4colheaderborder}%
    Put the glossary in a supertabular environment with four columns and a header
    bordered by horizontal lines and a horizontal line in the tail:
3988 \renewenvironment{theglossary}%
3989   {\tablehead{\hline
3990     \bfseries\entryname &
3991     \bfseries\descriptionname &
3992     \bfseries\symbolname &
3993     \bfseries\pagelistname\\ \hline}%
3994   \tabletail{\hline}%
3995   \begin{supertabular}%
3996     {\lllp{\glsgdescwidth}\lllp{\glspagelistwidth}}}%
3997   {\end{supertabular}}}%
3998 }

```

6.6 Glossary Styles using supertabular environment (glossary-superragged package)

The glossary styles defined in the package use the `supertabular` environment. These styles are like those provided by the package, except that the multiline columns

have ragged right justification.

```

3999 \ProvidesPackage{glossary-superragged}[2009/05/30 v2.01 (NLCT)]
    Requires the package:
4000 \RequirePackage{array}
    Requires the package:
4001 \RequirePackage{supertabular}

\glsdescwidth This is a length that governs the width of the description column. This may
                already have been defined.
4002 \@ifundefined{glsdescwidth}{%
4003   \newlength{glsdescwidth
4004   \setlength{glsdescwidth}{0.6\hsize}
4005 }{}

\glspagelistwidth This is a length that governs the width of the page list column. This may already
                   have been defined.
4006 \@ifundefined{glspagelistwidth}{%
4007   \newlength{glspagelistwidth
4008   \setlength{glspagelistwidth}{0.1\hsize}
4009 }{}

superragged The superragged glossary style uses the supertabular environment.
4010 \newglossarystyle{superragged}{%
    Put the glossary in a supertabular environment with two columns and no head or
    tail:
4011   \renewenvironment{theglossary}%
4012     {\tablehead{}\tabletail}%
4013     \begin{supertabular}{l>{\raggedright}p{glsdescwidth}}%
4014     {\end{supertabular}}%
    Do nothing at the start of the table:
4015   \renewcommand*{\glossaryheader}{}%
    No group headings:
4016   \renewcommand*{\glsgroupheading}[1]{}%
    Main (level 0) entries put in a row (name in first column, description and page
    list in second column):
4017   \renewcommand*{\glossaryentryfield}[5]{%
4018     \glstarget{##1}{##2} & ##3\glspostdescription\space ##5%
4019     \tabularnewline}%
    Sub entries put in a row (no name, description and page list in second column):
4020   \renewcommand*{\glossarysubentryfield}[6]{%
4021     & \glstarget{##2}{\strut}##4\glspostdescription\space ##6%
4022     \tabularnewline}%
    Blank row between groups:
4023   \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
4024 }
```

superraggedborder The `superraggedborder` style is like the above, but with horizontal and vertical lines:

```
4025 \newglossarystyle{superraggedborder}{%
    Base it on the glostylesuperragged style:
4026 \glossarystyle{superragged}%
    Put the glossary in a supertabular environment with two columns and a horizontal
    line in the head and tail:
4027 \renewenvironment{theglossary}%
4028 {\tablehead{\hline}\tabletail{\hline}%
4029 \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}}}%
4030 {\end{supertabular}}%
4031 }
```

superraggedheader The `superraggedheader` style is like the `super` style, but with a header:

```
4032 \newglossarystyle{superraggedheader}{%
    Base it on the glostylesuperragged style:
4033 \glossarystyle{superragged}%
    Put the glossary in a supertabular environment with two columns, a header and
    no tail:
4034 \renewenvironment{theglossary}%
4035 {\tablehead{\bfseries \entryname & \bfseries \descriptionname
4036 \tabularnewline}%
4037 \tabletail{}}%
4038 \begin{supertabular}{l>{\raggedright}p{\glsdescwidth}}}%
4039 {\end{supertabular}}%
4040 }
```

superraggedheaderborder The `superraggedheaderborder` style is like the `superragged` style but with a header and border:

```
4041 \newglossarystyle{superraggedheaderborder}{%
    Base it on the glostylesuper style:
4042 \glossarystyle{superragged}%
    Put the glossary in a supertabular environment with two columns, a header and
    horizontal lines above and below the table:
4043 \renewenvironment{theglossary}%
4044 {\tablehead{\hline\bfseries \entryname &
4045 \bfseries \descriptionname\tabularnewline\hline}%
4046 \tabletail{\hline}
4047 \begin{supertabular}{|l|>{\raggedright}p{\glsdescwidth}}}%
4048 {\end{supertabular}}%
4049 }
```

superragged3col The `superragged3col` style is like the `superragged` style, but with 3 columns:

```
4050 \newglossarystyle{superragged3col}{%
```

Put the glossary in a `supertabular` environment with three columns and no head or tail:

```
4051 \renewenvironment{theglossary}%
4052   {\tablehead{}\tabletail{}}%
4053   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}%
4054     >{\raggedright}p{\glspagelistwidth}}}%
4055   {\end{supertabular}}%
```

Do nothing at the start of the table:

```
4056 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
4057 \renewcommand*{\glsgroupeheading}[1]{}%
```

Main (level 0) entries on a row (name in first column, description in second column, page list in last column):

```
4058 \renewcommand*{\glossaryentryfield}[5]{%
4059   \glstarget{##1}{##2} & ##3 & ##5\tabularnewline}%
```

Sub entries on a row (no name, description in second column, page list in last column):

```
4060 \renewcommand*{\glossarysubentryfield}[6]{%
4061   & \glstarget{##2}{\strut}##4 & ##6\tabularnewline}%
```

Blank row between groups:

```
4062 \renewcommand*{\glsgroupskip}{ & \tabularnewline}%
4063 }
```

superragged3colborder The `superragged3colborder` style is like the `superragged3col` style, but with a border:

```
4064 \newglossarystyle{superragged3colborder}{%
```

Base it on the `glostylesuperragged3col` style:

```
4065 \glossarystyle{superragged3col}%
```

Put the glossary in a `supertabular` environment with three columns and a horizontal line in the head and tail:

```
4066 \renewenvironment{theglossary}%
4067   {\tablehead{\hline}\tabletail{\hline}%
4068   \begin{supertabular}{|l|>{\raggedright}p{\glsgdescwidth}|}%
4069     >{\raggedright}p{\glspagelistwidth}|}%
4070   {\end{supertabular}}%
4071 }
```

superragged3colheader The `superragged3colheader` style is like the `superragged3col` style but with a header row:

```
4072 \newglossarystyle{superragged3colheader}{%
```

Base it on the `glostylesuperragged3col` style:

```
4073 \glossarystyle{superragged3col}%
```

Put the glossary in a `supertabular` environment with three columns, a header and no tail:

```

4074 \renewenvironment{theglossary}%
4075   {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
4076     \bfseries\pagelistname\tabularnewline}\tabletail{}}%
4077   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}%
4078     >{\raggedright}p{\glspagelistwidth}}}%
4079   {\end{supertabular}}%
4080 }

```

`superraggedright3colheaderborder` The `superragged3colheaderborder` style is like the `superragged3col` style but with a header and border:

```

4081 \newglossarystyle{superragged3colheaderborder}{%
  Base it on the glostylesuperragged3colborder style:
4082 \glossarystyle{superragged3colborder}%
  Put the glossary in a supertabular environment with three columns, a header with
  horizontal lines and a horizontal line in the tail:
4083 \renewenvironment{theglossary}%
4084   {\tablehead{\hline
4085     \bfseries\entryname&\bfseries\descriptionname&
4086     \bfseries\pagelistname\tabularnewline\hline}%
4087   \tabletail{\hline}%
4088   \begin{supertabular}{l|>{\raggedright}p{\glsgdescwidth}|%
4089     >{\raggedright}p{\glspagelistwidth}}}%
4090   {\end{supertabular}}%
4091 }

```

`altsuperragged4col` The `altsuperragged4col` glossary style is like `altsuper4col` style in the package but uses ragged right formatting in the description and page list columns.

```

4092 \newglossarystyle{altsuperragged4col}{%
  Put the glossary in a supertabular environment with four columns and no head or
  tail:
4093 \renewenvironment{theglossary}%
4094   {\tablehead{}\tabletail{}}%
4095   \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}l%
4096     >{\raggedright}p{\glspagelistwidth}}}%
4097   {\end{supertabular}}%
  Do nothing at the start of the table:
4098 \renewcommand*{\glossaryheader}{}%
  No group headings:
4099 \renewcommand*{\glsgroupheading}[1]{}%
  Main (level 0) entries on a row with the name in the first column, description in
  second column, symbol in third column and page list in last column:
4100 \renewcommand*{\glossaryentryfield}[5]{%
4101   \glstarget{##1}{##2} & ##3 & ##4 & ##5\tabularnewline}%

```

Sub entries on a row with no name, the description in the second column, symbol in third column and page list in last column:

```
4102 \renewcommand*{\glossarysubentryfield}[6]{%
4103     & \glstarget{##2}{\strut}##4 & ##5 & ##6\tabularnewline}%

```

Blank row between groups:

```
4104 \renewcommand*{\glsgroupskip}{ & & \tabularnewline}%
4105 }

```

altsuperragged4colheader The altsuperragged4colheader style is like the altsuperragged4col style but with a header row.

```
4106 \newglossarystyle{altsuperragged4colheader}{%
    Base it on the glostylealtsuperragged4col style:
4107 \glossarystyle{altsuperragged4col}%
    Put the glossary in a supertabular environment with four columns, a header and
    no tail:
4108 \renewenvironment{theglossary}%
4109     {\tablehead{\bfseries\entryname&\bfseries\descriptionname&
4110         \bfseries\symbolname &
4111         \bfseries\pagelistname\tabularnewline}\tabletail{}}%
4112     \begin{supertabular}{l>{\raggedright}p{\glsgdescwidth}l%
4113         >{\raggedright}p{\glspagelistwidth}}}%
4114     {\end{supertabular}}}%
4115 }

```

altsuperragged4colborder The altsuperragged4colborder style is like the altsuperragged4col style but with a border.

```
4116 \newglossarystyle{altsuperragged4colborder}{%
    Base it on the glostylealtsuperragged4col style:
4117 \glossarystyle{altsuper4col}%
    Put the glossary in a supertabular environment with four columns and a horizontal
    line in the head and tail:
4118 \renewenvironment{theglossary}%
4119     {\tablehead{\hline}\tabletail{\hline}%
4120     \begin{supertabular}%
4121         {l|>{\raggedright}p{\glsgdescwidth}l|}%
4122         >{\raggedright}p{\glspagelistwidth}l}}}%
4123     {\end{supertabular}}}%
4124 }

```

altsuperragged4colheaderborder The altsuperragged4colheaderborder style is like the altsuperragged4col style but with a header and border.

```
4125 \newglossarystyle{altsuperragged4colheaderborder}{%
    Base it on the glostylealtsuperragged4col style:
4126 \glossarystyle{altsuperragged4col}%

```

Put the glossary in a `supertabular` environment with four columns and a header bordered by horizontal lines and a horizontal line in the tail:

```

4127 \renewenvironment{theglossary}%
4128   {\tablehead{\hline
4129     \bfseries\entryname &
4130     \bfseries\descriptionname &
4131     \bfseries\symbolname &
4132     \bfseries\pagelistname\tabularnewline\hline}%
4133   \tabletail{\hline}%
4134   \begin{supertabular}%
4135     {||>{\raggedright}p{\glsgdescwidth}|||}
4136     >{\raggedright}p{\glspagelistwidth}||}%
4137   {\end{supertabular}}%
4138 }

```

6.7 Tree Styles (glossary-tree.sty)

The style file defines glossary styles that have a tree-like structure. These are designed for hierarchical glossaries.

```

4139 \ProvidesPackage{glossary-tree}[2009/01/14 v1.01 (NLCT)]

```

index The index glossary style is similar in style to the way indices are usually typeset using `\item`, `\subitem` and `\subsubitem`. The entry name is set in bold. If an entry has a symbol, it is placed in brackets after the name. Then the description is displayed, followed by the number list. This style allows up to three levels.

```

4140 \newglossarystyle{index}{%

```

Set the paragraph indentation and skip and define `\item` to be the same as that used by `theindex`:

```

4141 \renewenvironment{theglossary}%
4142   {\setlength{\parindent}{0pt}%
4143   \setlength{\parskip}{0pt plus 0.3pt}%
4144   \let\item\@idxitem}%
4145   {%

```

Do nothing at the start of the environment:

```

4146 \renewcommand*{\glossaryheader}{}%

```

No group headers:

```

4147 \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entry starts a new item with the name in bold followed by the symbol in brackets (if it exists), the description and the page list.

```

4148 \renewcommand*{\glossaryentryfield}[5]{%
4149 \item\textbf{\glstarget{##1}{##2}}%
4150 \ifx\relax##4\relax
4151 \else
4152 \space##4%
4153 \fi
4154 \space##3\glspostdescription\space##5}%

```

Sub entries: level 1 entries use `\subitem`, levels greater than 1 use `\subsubitem`. The level (`##1`) shouldn't be 0, as that's catered by `\glossaryentryfield`, but for completeness, if the level is 0, `\item` is used. The name is put in bold, followed by the symbol in brackets (if it exists), the description and the page list.

```

4155 \renewcommand*{\glossarysubentryfield}[6]{%
4156     \ifcase##1\relax
4157         % level 0
4158         \item
4159     \or
4160         % level 1
4161         \subitem
4162     \else
4163         % all other levels
4164         \subsubitem
4165     \fi
4166     \textbf{\glstarget{##2}{##3}}%
4167     \ifx\relax##5\relax
4168     \else
4169         \space(##5)%
4170     \fi
4171     \space##4\glspostdescription\space ##6}%

```

Vertical gap between groups is the same as that used by indices:

```

4172 \renewcommand*{\glsgroupskip}{\indexspace}}

```

`indexgroup` The `indexgroup` style is like the `index` style but has headings.

```

4173 \newglossarystyle{indexgroup}{%

```

Base it on the `glostyleindex` style:

```

4174 \glossarystyle{index}%

```

Add a heading for each group. This puts the group's title in bold followed by a vertical gap.

```

4175 \renewcommand*{\glsgroupheading}[1]{%
4176     \item\textbf{\glsgetgrouptitle{##1}}\indexspace}%
4177 }

```

`indexhypergroup` The `indexhypergroup` style is like the `indexgroup` style but has hyper navigation.

```

4178 \newglossarystyle{indexhypergroup}{%

```

Base it on the `glostyleindex` style:

```

4179 \glossarystyle{index}%

```

Put navigation links to the groups at the start of the glossary:

```

4180 \renewcommand*{\glossaryheader}{%
4181     \item\textbf{\glsnavigation}\indexspace}%

```

Add a heading for each group (with a target). The group's title is in bold followed by a vertical gap.

```

4182 \renewcommand*{\glsgroupheading}[1]{%
4183     \item\textbf{\glsnavhypertarget{##1}{\glsgetgrouptitle{##1}}}%

```

```

4184     \indexspace}%
4185 }

```

tree The *tree* glossary style is similar in style to the *index* style, but can have arbitrary levels.

```

4186 \newglossarystyle{tree}{%

```

Set the paragraph indentation and skip:

```

4187   \renewenvironment{theglossary}%
4188     {\setlength{\parindent}{0pt}%
4189     \setlength{\parskip}{0pt plus 0.3pt}}%
4190     {}%

```

Do nothing at the start of the *theglossary* environment:

```

4191   \renewcommand*{\glossaryheader}{}%

```

No group headings:

```

4192   \renewcommand*{\glsgroupheading}[1]{}%

```

Main (level 0) entries: name in bold, followed by symbol in brackets (if it exists), the description and the page list:

```

4193   \renewcommand{\glossaryentryfield}[5]{%
4194     \hangindent0pt\relax
4195     \parindent0pt\relax
4196     \textbf{\glstarget{##1}{##2}}%
4197     \ifx\relax##4\relax
4198       \else
4199         \space{##4}%
4200       \fi
4201     \space ##3\glspostdescription \space ##5\par}%

```

Sub entries: level $\langle n \rangle$ is indented by $\langle n \rangle$ times `\glstreeindent`. The name is in bold, followed by the symbol in brackets (if it exists), the description and the page list.

```

4202   \renewcommand{\glossarysubentryfield}[6]{%
4203     \hangindent##1\glstreeindent\relax
4204     \parindent##1\glstreeindent\relax
4205     \textbf{\glstarget{##2}{##3}}%
4206     \ifx\relax##5\relax
4207       \else
4208         \space{##5}%
4209       \fi
4210     \space##4\glspostdescription\space ##6\par}%

```

Vertical gap between groups is the same as that used by indices:

```

4211   \renewcommand*{\glsgroupskip}{\indexspace}}

```

treegroup Like the *tree* style but the glossary groups have headings.

```

4212 \newglossarystyle{treegroup}{%

```

Base it on the *glostyletree* style:

```

4213   \glossarystyle{tree}%

```


Each group has a heading (in bold) followed by a vertical gap):

```
4214 \renewcommand{\glsgroupheading}[1]{\par
4215 \noindent\textbf{\glsgrouptitle{##1}}\par\indexspace}%
4216 }
```

treehypergroup The **treehypergroup** style is like the **treegroup** style, but has a set of links to the groups at the start of the glossary.

```
4217 \newglossarystyle{treehypergroup}{%
```

Base it on the **glostyletree** style:

```
4218 \glossarystyle{tree}%
```

Put navigation links to the groups at the start of the **theglossary** environment:

```
4219 \renewcommand*{\glossaryheader}{%
4220 \par\noindent\textbf{\glsnavigation}\par\indexspace}%
```

Each group has a heading (in bold with a target) followed by a vertical gap):

```
4221 \renewcommand*{\glsgroupheading}[1]{%
4222 \par\noindent
4223 \textbf{\glsnavigationhypertarget{##1}{\glsgrouptitle{##1}}}\par
4224 \indexspace}%
4225 }
```

\glstreeindent Length governing left indent for each level of the **tree** style.

```
4226 \newlength\glstreeindent
4227 \setlength{\glstreeindent}{10pt}
```

treenoname The **treenoname** glossary style is like the **tree** style, but doesn't print the name or symbol for sub-levels.

```
4228 \newglossarystyle{treenoname}{%
```

Set the paragraph indentation and skip:

```
4229 \renewenvironment{theglossary}%
4230 {\setlength{\parindent}{0pt}%
4231 \setlength{\parskip}{0pt plus 0.3pt}}%
4232 {}%
```

No header:

```
4233 \renewcommand*{\glossaryheader}{}%
```

No group headings:

```
4234 \renewcommand*{\glsgroupheading}[1]{}%
```

Main (level 0) entries: the name is in bold, followed by the symbol in brackets (if it exists), the description and the page list.

```
4235 \renewcommand{\glossaryentryfield}[5]{%
4236 \hangindent0pt\relax
4237 \parindent0pt\relax
4238 \textbf{\glstarget{##1}{##2}}%
4239 \ifx\relax##4\relax
4240 \else
```

```

4241     \space{##4}%
4242     \fi
4243     \space ##3\glspostdescription \space ##5\par}%

Sub entries: level  $\langle n \rangle$  is indented by  $\langle n \rangle$  times \glstreeindent. The name and
symbol are omitted. The description followed by the page list are displayed.

4244 \renewcommand{\glossarysubentryfield}[6]{%
4245     \hangindent##1\glstreeindent\relax
4246     \parindent##1\glstreeindent\relax
4247     \glstarget{##2}{\strut}%
4248     ##4\glspostdescription\space ##6\par}%

Vertical gap between groups is the same as that used by indices:

4249 \renewcommand*{\glsgroupskip}{\indexspace}%
4250 }
```

treenonamegroup Like the `treenoname` style but the glossary groups have headings.

```

4251 \newglossarystyle{treenonamegroup}{%
    Base it on the glostyletreenoname style:
4252 \glossarystyle{treenoname}%
    Give each group a heading:
4253 \renewcommand{\glsgroupheading}[1]{\par
4254     \noindent\textbf{\glsgrouptitle{##1}}\par\indexspace}%
4255 }
```

treenonamehypergroup The `treenonamehypergroup` style is like the `treenonamegroup` style, but has a set of links to the groups at the start of the glossary.

```

4256 \newglossarystyle{treenonamehypergroup}{%
    Base it on the glostyletreenoname style:
4257 \glossarystyle{treenoname}%
    Put navigation links to the groups at the start of the theglossary environment:
4258 \renewcommand*{\glossaryheader}{%
4259     \par\noindent\textbf{\glsnavigation}\par\indexspace}%
    Each group has a heading (in bold with a target) followed by a vertical gap):
4260 \renewcommand*{\glsgroupheading}[1]{%
4261     \par\noindent
4262     \textbf{\glsnavigationtarget{##1}{\glsgrouptitle{##1}}}\par
4263     \indexspace}%
4264 }
```

\glsssetwidest `\glsssetwidest[$\langle level \rangle$]{ $\langle text \rangle$ }` sets the widest text for the given level. It is used by the `alttree` glossary styles to determine the indentation of each level.

```

4265 \newcommand*{\glsssetwidest}[2][0]{%
4266     \expandafter\def\csname @glswidestname\romannumeral#1\endcsname{%
4267         #2}%
4268 }
```

```

\@glswidestname Initialise \@glswidestname.
4269 \newcommand*{\@glswidestname}{\}

almtree The almtree glossary style is similar in style to the tree style, but the indentation is
        obtained from the width of \@glswidestname which is set using \glsetwidest.
4270 \newglossarystyle{almtree}{%
    Redefine theglossary environment.
4271 \renewenvironment{theglossary}%
4272     {\def\@gls@prevlevel{-1}%
4273      \mbox{}\par}%
4274     {\par}%

    Set the header and group headers to nothing.
4275 \renewcommand*{\glossaryheader}{\}%
4276 \renewcommand*{\glsgroupheading}[1]{\}%

    Redefine the way that the level 0 entries are displayed.
4277 \renewcommand{\glossaryentryfield}[5]{%

        If the level hasn't changed, keep the same settings, otherwise change \glstreeindent
        accordingly.
4278     \ifnum\@gls@prevlevel=0\relax
4279     \else

        Find out how big the indentation should be by measuring the widest entry.
4280         \settowidth{\glstreeindent}{\textbf{\@glswidestname\space}}%

        Set the hangindent and paragraph indent.
4281         \hangindent\glstreeindent
4282         \parindent\glstreeindent
4283         \fi

        Put the name to the left of the paragraph block.
4284         \makebox[0pt][r]{\makebox[\glstreeindent][l]{%
4285             \textbf{\glstarget{##1}{##2}}}%

            If the symbol is missing, ignore it, otherwise put it in brackets.
4286         \ifx\relax##4\relax
4287         \else
4288             (##4)\space
4289         \fi

        Do the description followed by the description terminator and location list.
4290         ##3\glspostdescription \space ##5\par

        Set the previous level to 0.
4291         \def\@gls@prevlevel{0}%
4292     }%

    Redefine the way sub-entries are displayed.
4293 \renewcommand{\glossarysubentryfield}[6]{%

```

If the level hasn't changed, keep the same settings, otherwise adjust `\glstreeindent` accordingly.

```
4294     \ifnum\@gls@prevlevel=##1\relax
4295     \else
```

Compute the widest entry for this level, or for level 0 if not defined for this level.
Store in `\gls@tmplen`

```
4296     \ifundefined{@glswidestname\romannumeral##1}{%
4297     \settowidth{\gls@tmplen}{\textbf{@glswidestname\space}}{%
4298     \settowidth{\gls@tmplen}{\textbf{%
4299     \csname @glswidestname\romannumeral##1\endcsname\space}}}%
```

Determine if going up or down a level

```
4300     \ifnum\@gls@prevlevel<##1\relax
```

Depth has increased, so add the width of the widest entry to `\glstreeindent`.

```
4301     \setlength\glstreeindent\gls@tmplen
4302     \addtolength\glstreeindent\parindent
4303     \parindent\glstreeindent
4304     \else
```

Depth has decreased, so subtract width of the widest entry from the previous level to `\glstreeindent`. First determine the width of the widest entry for the previous level and store in `\glstreeindent`.

```
4305     \ifundefined{@glswidestname\romannumeral\@gls@prevlevel}{%
4306     \settowidth{\glstreeindent}{\textbf{%
4307     \@glswidestname\space}}{%
4308     \settowidth{\glstreeindent}{\textbf{%
4309     \csname @glswidestname\romannumeral\@gls@prevlevel
4310     \endcsname\space}}}%
```

Subtract this length from the previous level's paragraph indent and set to `\glstreeindent`.

```
4311     \addtolength\parindent{-\glstreeindent}%
4312     \setlength\glstreeindent\parindent
4313     \fi
4314     \fi
```

Set the hanging indentation.

```
4315     \hangindent\glstreeindent
```

Put the name to the left of the paragraph block

```
4316     \makebox[0pt][r]{\makebox[\gls@tmplen][l]{%
4317     \textbf{\glstarget{##2}{##3}}}%
```

If the symbol is missing, ignore it, otherwise put it in brackets.

```
4318     \ifx##5\relax\relax
4319     \else
4320     (##5)\space
4321     \fi
```

Do the description followed by the description terminator and location list.

```
4322     ##4\glspostdescription\space ##6\par
```

Set the previous level macro to the current level.

```
4323 \def\@gls@prevlevel{##1}%
4324 }%
```

Vertical gap between groups is the same as that used by indices:

```
4325 \renewcommand*{\glsgroupskip}{\indexspace}%
4326 }
```

almtreegroup Like the `almtree` style but the glossary groups have headings.

```
4327 \newglossarystyle{almtreegroup}{%
Base it on the glostylealmtree style:
4328 \glossarystyle{almtree}%
Give each group a heading.
4329 \renewcommand{\glsgroupheading}[1]{\par
4330 \def\@gls@prevlevel{-1}%
4331 \hangindent0pt\relax
4332 \parindent0pt\relax
4333 \textbf{\glsgetgrouptitle{##1}}\par\indexspace}%
4334 }
```

almtreehypergroup The `almtreehypergroup` style is like the `almtreegroup` style, but has a set of links to the groups at the start of the glossary.

```
4335 \newglossarystyle{almtreehypergroup}{%
Base it on the glostylealmtree style:
4336 \glossarystyle{almtree}%
Put the navigation links in the header
4337 \renewcommand*{\glossaryheader}{%
4338 \par
4339 \def\@gls@prevlevel{-1}%
4340 \hangindent0pt\relax
4341 \parindent0pt\relax
4342 \textbf{\glsnavigation}\par\indexspace}%
Put a hypertarget at the start of each group
4343 \renewcommand*{\glsgroupheading}[1]{%
4344 \par
4345 \def\@gls@prevlevel{-1}%
4346 \hangindent0pt\relax
4347 \parindent0pt\relax
4348 \textbf{\glsnavhypertarget{##1}}{\glsgetgrouptitle{##1}}\par
4349 \indexspace}}
```

7 Accessibilty Support (glossaries-accsupp Code)

The package is experimental. It is intended to provide a means of using the PDF accessibilty support in glossary entries. See the documentation for further details about accessibility support.

```

4350 \NeedsTeXFormat{LaTeX2e}
4351 \ProvidesPackage{glossaries-accsupp}[2009/11/02 v0.2 (NLCT)]
    Pass all options to glossaries:
4352 \DeclareOption*{\PassOptionsToPackage{\CurrentOption}{glossaries}}
    Process options:
4353 \ProcessOptions
    Required packages:
4354 \RequirePackage{glossaries}
4355 \RequirePackage{accsupp}

```

7.1 Defining Replacement Text

The version 0.1 stored the replacement text in the `symbol` key. This has been changed to use the new keys defined here. Example of use:

```
\newglossaryentry{dr}{name=Dr,description={},access={Doctor}}
```

access The replacement text corresponding to the `name` key:

```

4356 \define@key{glossentry}{access}{%
4357   \def\@glo@access{#1}%
4358 }

```

textaccess The replacement text corresponding to the `text` key:

```

4359 \define@key{glossentry}{textaccess}{%
4360   \def\@glo@textaccess{#1}%
4361 }

```

firstaccess The replacement text corresponding to the `first` key:

```

4362 \define@key{glossentry}{firstaccess}{%
4363   \def\@glo@firstaccess{#1}%
4364 }

```

pluralaccess The replacement text corresponding to the `plural` key:

```

4365 \define@key{glossentry}{pluralaccess}{%
4366   \def\@glo@pluralaccess{#1}%
4367 }

```

firstpluralaccess The replacement text corresponding to the `firstplural` key:

```

4368 \define@key{glossentry}{firstpluralaccess}{%
4369   \def\@glo@firstpluralaccess{#1}%
4370 }

```

symbolaccess The replacement text corresponding to the `symbol` key:

```

4371 \define@key{glossentry}{symbolaccess}{%
4372   \def\@glo@symbolaccess{#1}%
4373 }

```

symbolpluralaccess The replacement text corresponding to the symbolplural key:

```
4374 \define@key{glossentry}{symbolpluralaccess}{%
4375   \def\@glo@symbolpluralaccess{#1}%
4376 }
```

descriptionaccess The replacement text corresponding to the description key:

```
4377 \define@key{glossentry}{descriptionaccess}{%
4378   \def\@glo@descaccess{#1}%
4379 }
```

descriptionpluralaccess The replacement text corresponding to the descriptionplural key:

```
4380 \define@key{glossentry}{descriptionpluralaccess}{%
4381   \def\@glo@descpluralaccess{#1}%
4382 }
```

There are no equivalent keys for the user1...user6 keys. The replacement text would have to be explicitly put in the value, e.g., user1={\glsaccsupp{inches}{in}}.

\@gls@noaccess Indicates that no replacement text has been provided.

```
4383 \def\@gls@noaccess{\relax}
```

Add to the start hook (the access key is initialised to the value of the symbol key at the start for backwards compatibility):

```
4384 \let\@gls@oldnewglossaryentryprehook\@newglossaryentryprehook
4385 \renewcommand*{\@newglossaryentryprehook}{%
4386   \@gls@oldnewglossaryentryprehook
4387   \def\@glo@access{\@glo@symbol}%
4388 }
```

Initialise the other keys:

```
4388 \def\@glo@textaccess{\@glo@access}%
4389 \def\@glo@firstaccess{\@glo@access}%
4390 \def\@glo@pluralaccess{\@glo@textaccess}%
4391 \def\@glo@firstpluralaccess{\@glo@pluralaccess}%
4392 \def\@glo@symbolaccess{\relax}%
4393 \def\@glo@symbolpluralaccess{\@glo@symbolaccess}%
4394 \def\@glo@descaccess{\relax}%
4395 \def\@glo@descpluralaccess{\@glo@descaccess}%
4396 }
```

Add to the end hook:

```
4397 \let\@gls@oldnewglossaryentryposthook\@newglossaryentryposthook
4398 \renewcommand*{\@newglossaryentryposthook}{%
4399   \@gls@oldnewglossaryentryposthook
4400 }
```

Store the access information:

```
4400 \expandafter
4401   \protected@xdef\csname glo@\@glo@label @access\endcsname{%
4402     \@glo@access}%
4403 \expandafter
4404   \protected@xdef\csname glo@\@glo@label @textaccess\endcsname{%
```

```

4405     \@glo@textaccess}%
4406 \expandafter
4407     \protected@xdef\csname glo@\@glo@label @firstaccess\endcsname{%
4408         \@glo@firstaccess}%
4409 \expandafter
4410     \protected@xdef\csname glo@\@glo@label @pluralaccess\endcsname{%
4411         \@glo@pluralaccess}%
4412 \expandafter
4413     \protected@xdef\csname glo@\@glo@label @firstpluralaccess\endcsname{%
4414         \@glo@firstpluralaccess}%
4415 \expandafter
4416     \protected@xdef\csname glo@\@glo@label @symbolaccess\endcsname{%
4417         \@glo@symbolaccess}%
4418 \expandafter
4419     \protected@xdef\csname glo@\@glo@label @symbolpluralaccess\endcsname{%
4420         \@glo@symbolpluralaccess}%
4421 \expandafter
4422     \protected@xdef\csname glo@\@glo@label @descaccess\endcsname{%
4423         \@glo@descaccess}%
4424 \expandafter
4425     \protected@xdef\csname glo@\@glo@label @descpluralaccess\endcsname{%
4426         \@glo@descpluralaccess}%
4427 }

```

7.2 Accessing Replacement Text

`\glsentryaccess` Get the value of the access key for the entry with the given label:

```

4428 \newcommand*{\glsentryaccess}[1]{%
4429     \csname glo@#1@access\endcsname
4430 }

```

`\glsentrytextaccess` Get the value of the textaccess key for the entry with the given label:

```

4431 \newcommand*{\glsentrytextaccess}[1]{%
4432     \csname glo@#1@textaccess\endcsname
4433 }

```

`\glsentryfirstaccess` Get the value of the firstaccess key for the entry with the given label:

```

4434 \newcommand*{\glsentryfirstaccess}[1]{%
4435     \csname glo@#1@firstaccess\endcsname
4436 }

```

`\glsentrypluralaccess` Get the value of the pluralaccess key for the entry with the given label:

```

4437 \newcommand*{\glsentrypluralaccess}[1]{%
4438     \csname glo@#1@pluralaccess\endcsname
4439 }

```

`\glsentryfirstpluralaccess` Get the value of the firstpluralaccess key for the entry with the given label:

```

4440 \newcommand*{\glsentryfirstpluralaccess}[1]{%
4441     \csname glo@#1@firstpluralaccess\endcsname
4442 }

```



```

\glsentrysymbolaccess  Get the value of the symbolaccess key for the entry with the given label:
4443 \newcommand*{\glsentrysymbolaccess}[1]{%
4444   \csname glo@#1@symbolaccess\endcsname
4445 }

\glsentrysymbolpluralaccess  Get the value of the symbolpluralaccess key for the entry with the given label:
4446 \newcommand*{\glsentrysymbolpluralaccess}[1]{%
4447   \csname glo@#1@symbolpluralaccess\endcsname
4448 }

\glsentrydescaccess  Get the value of the descriptionaccess key for the entry with the given label:
4449 \newcommand*{\glsentrydescaccess}[1]{%
4450   \csname glo@#1@descaccess\endcsname
4451 }

\glsentrydescpluralaccess  Get the value of the descriptionpluralaccess key for the entry with the given label:
4452 \newcommand*{\glsentrydescpluralaccess}[1]{%
4453   \csname glo@#1@descaccess\endcsname
4454 }

\glsaccsupp  \glsaccsupp{<replacement text>}{<text>}

This can be redefined to use E or Alt instead of ActualText. (I don't have the
software to test the E or Alt options.)
4455 \newcommand*{\glsaccsupp}[2]{%
4456   \BeginAccSupp{ActualText=#1}#2\EndAccSupp{}}%
4457 }

\xglsaccsupp  Fully expands replacement text before calling \glsaccsupp
4458 \newcommand*{\xglsaccsupp}[2]{%
4459   \protected@edef\@gls@replacementtext{#1}%
4460   \expandafter\glsaccsupp\expandafter{\@gls@replacementtext}{#2}%
4461 }

\glsnameaccessdisplay  Displays the first argument with the accessibility text for the entry with the label
given by the second argument (if set).
4462 \DeclareRobustCommand*{\glsnameaccessdisplay}[2]{%
4463   \protected@edef\@glo@access{\glsentryaccess{#2}}%
4464   \ifx\@glo@access\@gls@noaccess
4465     #1%
4466   \else
4467     \xglsaccsupp{\@glo@access}{#1}%
4468   \fi
4469 }

\glstextaccessdisplay  As above but for the textaccess replacement text.
4470 \DeclareRobustCommand*{\glstextaccessdisplay}[2]{%
4471   \protected@edef\@glo@access{\glsentrytextaccess{#2}}%

```

```

4472 \ifx\@glo@access\@gls@noaccess
4473   #1%
4474 \else
4475   \xglsaccsupp{\@glo@access}{#1}%
4476 \fi
4477 }

```

`\glspluralaccessdisplay` As above but for the pluralaccess replacement text.

```

4478 \DeclareRobustCommand*\glspluralaccessdisplay}[2]{%
4479   \protected@edef\@glo@access{\glsentrypluralaccess{#2}}%
4480   \ifx\@glo@access\@gls@noaccess
4481     #1%
4482   \else
4483     \xglsaccsupp{\@glo@access}{#1}%
4484   \fi
4485 }

```

`\glsfirstaccessdisplay` As above but for the firstaccess replacement text.

```

4486 \DeclareRobustCommand*\glsfirstaccessdisplay}[2]{%
4487   \protected@edef\@glo@access{\glsentryfirstaccess{#2}}%
4488   \ifx\@glo@access\@gls@noaccess
4489     #1%
4490   \else
4491     \xglsaccsupp{\@glo@access}{#1}%
4492   \fi
4493 }

```

`\glsfirstpluralaccessdisplay` As above but for the firstpluralaccess replacement text.

```

4494 \DeclareRobustCommand*\glsfirstpluralaccessdisplay}[2]{%
4495   \protected@edef\@glo@access{\glsentryfirstpluralaccess{#2}}%
4496   \ifx\@glo@access\@gls@noaccess
4497     #1%
4498   \else
4499     \xglsaccsupp{\@glo@access}{#1}%
4500   \fi
4501 }

```

`\glssymbolaccessdisplay` As above but for the symbolaccess replacement text.

```

4502 \DeclareRobustCommand*\glssymbolaccessdisplay}[2]{%
4503   \protected@edef\@glo@access{\glsentrysymbolaccess{#2}}%
4504   \ifx\@glo@access\@gls@noaccess
4505     #1%
4506   \else
4507     \xglsaccsupp{\@glo@access}{#1}%
4508   \fi
4509 }

```

`\glssymbolpluralaccessdisplay` As above but for the symbolpluralaccess replacement text.

```

4510 \DeclareRobustCommand*\glssymbolpluralaccessdisplay}[2]{%

```

```

4511 \protected@edef\@glo@access{\glsentrysymbolpluralaccess{#2}}%
4512 \ifx\@glo@access\@gls@noaccess
4513   #1%
4514 \else
4515   \xglsaccsupp{\@glo@access}{#1}%
4516 \fi
4517 }

```

`\glsdescriptionaccessdisplay` As above but for the descriptionaccess replacement text.

```

4518 \DeclareRobustCommand*\glsdescriptionaccessdisplay}[2]{%
4519 \protected@edef\@glo@access{\glsentrydescaccess{#2}}%
4520 \ifx\@glo@access\@gls@noaccess
4521   #1%
4522 \else
4523   \xglsaccsupp{\@glo@access}{#1}%
4524 \fi
4525 }

```

`\glsdescriptionpluralaccessdisplay` As above but for the descriptionpluralaccess replacement text.

```

4526 \DeclareRobustCommand*\glsdescriptionpluralaccessdisplay}[2]{%
4527 \protected@edef\@glo@access{\glsentrydesclpluralaccess{#2}}%
4528 \ifx\@glo@access\@gls@noaccess
4529   #1%
4530 \else
4531   \xglsaccsupp{\@glo@access}{#1}%
4532 \fi
4533 }

```

`\glsaccessdisplay` Gets the replacement text corresponding to the named key given by the first argument and calls the appropriate command defined above.

```

4534 \DeclareRobustCommand*\glsaccessdisplay}[3]{%
4535 \@ifundefined{gls#1accessdisplay}%
4536 {%
4537   \PackageError{glossaries-accsupp}{No accessibility support
4538     for key ‘#1’}{}%
4539 }%
4540 {%
4541   \csname gls#1accessdisplay\endcsname{#2}{#3}%
4542 }%
4543 }

```

`\@gls@` Redefine `\@gls@` to change the way the link text is defined

```

4544 \def\@gls@#1#2[#3]{%
4545 \glsdoifexists{#2}%
4546 {%
4547   \edef\@glo@type{\glsentrytype{#2}}%
4548   Save options in \@gls@link@opts and label in \@gls@link@label
4549   \def\@gls@link@opts{#1}%
4549   \def\@gls@link@label{#2}%

```

Determine what the link text should be (this is stored in `\@glo@text`). This is no longer expanded.

```

4550 \ifglsused{#2}%
4551 {%
4552 \def\@glo@text{\csname gls@\@glo@type @display\endcsname
4553 {\glstextaccessdisplay{\glentrytext{#2}}{#2}}%
4554 {\glsdescriptionaccessdisplay{\glentrydesc{#2}}{#2}}%
4555 {\glsymbolaccessdisplay{\glentrysymbol{#2}}{#2}}%
4556 {#3}}%
4557 }%
4558 {%
4559 \def\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
4560 {\glsfirstaccessdisplay{\glentryfirst{#2}}{#2}}%
4561 {\glsdescriptionaccessdisplay{\glentrydesc{#2}}{#2}}%
4562 {\glsymbolaccessdisplay{\glentrysymbol{#2}}{#2}}%
4563 {#3}}%
4564 }%

```

Call `\@gls@link`. If footnote package option has been used, suppress hyperlink for first use.

```

4565 \ifglsused{#2}%
4566 {%
4567 \@gls@link[#1]{#2}{\@glo@text}%
4568 }%
4569 {%
4570 \gls@checkisacronymlist\@glo@type
4571 \ifthenelse{(\boolean{@glsisacronymlist})\AND
4572 \boolean{glsacrfootnote}) \OR \NOT\boolean{glshyperfirst}}%
4573 {%
4574 \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
4575 }%
4576 {%
4577 \@gls@link[#1]{#2}{\@glo@text}%
4578 }%
4579 }%

```

Indicate that this entry has now been used

```

4580 \glsunset{#2}%
4581 }%
4582 }

```

`\@Gls@`

```

4583 \def\@Gls@#1#2[#3]{%
4584 \glsdoifexists{#2}%
4585 {%
4586 \edef\@glo@type{\glentrytype{#2}}%
4587 \def\@gls@link@opts{#1}%
4588 \def\@gls@link@label{#2}%

```

Save options in `\@gls@link@opts` and label in `\@gls@link@label`

Determine what the link text should be (this is stored in `\@glo@text`). The first character of the entry text is converted to uppercase before passing to `\gls@<type>@display` or `\gls@<type>@displayfirst`

```

4589     \ifglsused{#2}%
4590     {%
4591         \def\@glo@text{\csname gls@\@glo@type @display\endcsname
4592             {\glstextaccessdisplay{\Glsentrytext{#2}}{#2}}%
4593             {\glsdescriptionaccessdisplay{\glentrydesc{#2}}{#2}}%
4594             {\glssymbolaccessdisplay{\glentrysymbol{#2}}{#2}}%
4595             {#3}}%
4596     }%
4597     {%
4598         \def\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
4599             {\glsfirstaccessdisplay{\Glsentryfirst{#2}}{#2}}%
4600             {\glsdescriptionaccessdisplay{\glentrydesc{#2}}{#2}}%
4601             {\glssymbolaccessdisplay{\glentrysymbol{#2}}{#2}}%
4602             {#3}}%
4603     }%

```

Call `\@gls@link`. If footnote package option has been used, suppress hyperlink for first use.

```

4604     \ifglsused{#2}%
4605     {%
4606         \@gls@link[#1]{#2}{\@glo@text}%
4607     }%
4608     {%
4609         \gls@checkisacronymlist\@glo@type
4610         \ifthenelse{\boolean{@glsisacronymlist}}\AND
4611             \boolean{glsacrfootnote}\) \OR\nOT\boolean{glshyperfirst}}%
4612     {%
4613         \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
4614     }%
4615     {%
4616         \@gls@link[#1]{#2}{\@glo@text}%
4617     }%
4618 }%

```

Indicate that this entry has now been used

```

4619     \glsunset{#2}%
4620 }%
4621 }

```

`\@GLS@`

```

4622 \def\@GLS@#1#2[#3]{%
4623     \glsdoifexists{#2}{%
4624         \edef\@glo@type{\glentrytype{#2}}%

```

Save options in `\@gls@link@opts` and label in `\@gls@link@label`

```

4625     \def\@gls@link@opts{#1}%
4626     \def\@gls@link@label{#2}%

```

Determine what the link text should be (this is stored in \@glo@text).

```

4627 \ifglsused{#2}%
4628 {%
4629 \def\@glo@text{\csname gls@\@glo@type @display\endcsname
4630 {\glstextaccessdisplay{\glentrytext{#2}}{#2}}}%
4631 {\glsdescriptionaccessdisplay{\glentrydesc{#2}}{#2}}}%
4632 {\glsymbolaccessdisplay{\glentrysymbol{#2}}{#2}}}%
4633 {#3}}%
4634 }%
4635 {%
4636 \edef\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
4637 {\glsfirstaccessdisplay{\glentryfirst{#2}}{#2}}}%
4638 {\glsdescriptionaccessdisplay{\glentrydesc{#2}}{#2}}}%
4639 {\glsymbolaccessdisplay{\glentrysymbol{#2}}{#2}}}%
4640 {#3}}%
4641 }%

```

Call \@gls@link If footnote package option has been used, suppress hyperlink for first use.

```

4642 \ifglsused{#2}%
4643 {%
4644 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
4645 }%
4646 {%
4647 \gls@checkisacronymlist\@glo@type
4648 \ifthenelse{(\boolean{@glsisacronymlist})\AND
4649 \boolean{glsacrfootnote})\OR\NOT\boolean{glshyperfirst}}{%
4650 \@gls@link[#1,hyper=false]{#2}{\MakeUppercase{\@glo@text}}%
4651 }%
4652 {%
4653 \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
4654 }%
4655 }%

```

Indicate that this entry has now been used

```

4656 \glsunset{#2}%
4657 }%
4658 }

```

\@gls@pl@

```

4659 \def\@glspl@#1#2[#3]{%
4660 \glsdoifexists{#2}%
4661 {%
4662 \edef\@glo@type{\glentrytype{#2}}%

```

Save options in \@gls@link@opts and label in \@gls@link@label

```

4663 \def\@gls@link@opts{#1}%
4664 \def\@gls@link@label{#2}%

```

Determine what the link text should be (this is stored in \@glo@text)

```

4665 \ifglsused{#2}%

```

```

4666   {%
4667   \def\@glo@text{\csname gls@\@glo@type @display\endcsname
4668   {\glspluralaccessdisplay{\glsentryplural{#2}}{#2}}%
4669   {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
4670   {\glsymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
4671   {#3}}%
4672   }%
4673   {%
4674   \def\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
4675   {\glsfirstpluralaccessdisplay{\glsentryfirstplural{#2}}{#2}}%
4676   {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
4677   {\glsymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
4678   {#3}}%
4679   }%

```

Call `\@gls@link` If footnote package option has been used, suppress hyperlink for first use.

```

4680   \ifglsused{#2}%
4681   {%
4682   \@gls@link[#1]{#2}{\@glo@text}%
4683   }%
4684   {%
4685   \gls@checkisacronymlist\@glo@type
4686   \ifthenelse{(\boolean{@glsisacronymlist})\AND
4687   \boolean{glsacrfootnote}}{\OR\NOT\boolean{glshyperfirst}}%
4688   {%
4689   \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
4690   }%
4691   {%
4692   \@gls@link[#1]{#2}{\@glo@text}%
4693   }%
4694   }%

```

Indicate that this entry has now been used

```

4695   \glsunset{#2}%
4696   }%
4697 }

```

`\@Glspl@`

```

4698 \def\@Glspl@#1#2[#3]{%
4699 \glsdoifexists{#2}%
4700 {%
4701   \edef\@glo@type{\glsentrytype{#2}}%

```

Save options in `\@gls@link@opts` and label in `\@gls@link@label`

```

4702   \def\@gls@link@opts{#1}%
4703   \def\@gls@link@label{#2}%

```

Determine what the link text should be (this is stored in `\@glo@text`).

```

4704   \ifglsused{#2}%
4705   {%

```

```

4706 \def\@glo@text{\csname gls@\@glo@type @display\endcsname
4707 {\glspluralaccessdisplay{\Glsentryplural{#2}}{#2}}%
4708 {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
4709 {\glsymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
4710 {#3}}%
4711 }%
4712 {%
4713 \def\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
4714 {\glsfirstpluralaccessdisplay{\Glsentryfirstplural{#2}}{#2}}%
4715 {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
4716 {\glsymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
4717 {#3}}%
4718 }%

```

Call \@gls@link If footnote package option has been used, suppress hyperlink for first use.

```

4719 \ifglsused{#2}%
4720 {%
4721 \@gls@link[#1]{#2}{\@glo@text}%
4722 }%
4723 {%
4724 \ifthenelse{\equal{\@glo@type}{\acronymtype}\and
4725 \boolean{glsacrfootnote}}%
4726 {%
4727 \@gls@link[#1,hyper=false]{#2}{\@glo@text}%
4728 }%
4729 {%
4730 \@gls@link[#1]{#2}{\@glo@text}%
4731 }%
4732 }%

```

Indicate that this entry has now been used

```

4733 \glsunset{#2}%
4734 }%
4735 }

```

\@GLSp1@

```

4736 \def\@GLSp1@#1#2[#3]{%
4737 \glsdoifexists{#2}%
4738 {%
4739 \edef\@glo@type{\glsentrytype{#2}}%

```

Save options in \@gls@link@opts and label in \@gls@link@label

```

4740 \def\@gls@link@opts{#1}%
4741 \def\@gls@link@label{#2}%

```

Determine what the link text should be (this is stored in \@glo@text)

```

4742 \ifglsused{#2}%
4743 {%
4744 \def\@glo@text{\csname gls@\@glo@type @display\endcsname
4745 {\glspluralaccessdisplay{\glsentryplural{#2}}{#2}}%

```



```

4746      {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
4747      {\glssymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
4748      {#3}}%
4749  }%
4750  {%
4751      \def\@glo@text{\csname gls@\@glo@type @displayfirst\endcsname
4752      {\glsfirstpluralaccessdisplay{\glsentryfirstplural{#2}}{#2}}%
4753      {\glsdescriptionpluralaccessdisplay{\glsentrydescplural{#2}}{#2}}%
4754      {\glssymbolpluralaccessdisplay{\glsentrysymbolplural{#2}}{#2}}%
4755      {#3}}%
4756  }%

```

Call `\@gls@link` If footnote package option has been used, suppress hyperlink for first use.

```

4757      \ifglsused{#2}%
4758      {%
4759          \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
4760      }%
4761      {%
4762          \gls@checkisacronymlist\@glo@type
4763          \ifthenelse{(\boolean{glsisacronymlist})\AND
4764          \boolean{glsacrfootnote}}{\OR\NOT\boolean{glshyperfirst}}%
4765          {%
4766              \@gls@link[#1,hyper=false]{#2}{\MakeUppercase{\@glo@text}}%
4767          }%
4768          {%
4769              \@gls@link[#1]{#2}{\MakeUppercase{\@glo@text}}%
4770          }%
4771      }%

```

Indicate that this entry has now been used

```

4772      \glsunset{#2}%
4773  }%
4774 }

```

7.3 Displaying the Glossary

Entries within the glossary or list of acronyms are now formatted via `\accsuppglossaryentryfield` and `\accsuppglossarysubentryfield`.

`\@glossaryentryfield`

```

4775 \ifglsxindy
4776   \renewcommand*{\@glossaryentryfield}{%
4777       \string\accsuppglossaryentryfield}
4778 \else
4779   \renewcommand*{\@glossaryentryfield}{%
4780       \string\accsuppglossaryentryfield}
4781 \fi

```

`\@glossarysubentryfield`

```

4782 \ifglxindy
4783   \renewcommand*{\@glossarysubentryfield}{%
4784     \string\accsuppglossarysubentryfield}
4785 \else
4786   \renewcommand*{\@glossarysubentryfield}{%
4787     \string\accsuppglossarysubentryfield}
4788 \fi

```

\accsuppglossaryentryfield

```

4789 \newcommand*{\accsuppglossaryentryfield}[5]{%
4790   \glossaryentryfield{#1}{#2}%
4791   {\glsglssymbolaccessdisplay{#2}{#1}}%
4792   {\glsglssymbolaccessdisplay{#3}{#1}}%
4793   {\glsglssymbolaccessdisplay{#4}{#1}}{#5}%
4794 }

```

\accsuppglossarysubentryfield

```

4795 \newcommand*{\accsuppglossarysubentryfield}[6]{%
4796   \glossaryentryfield{#1}{#2}%
4797   {\glsglssymbolaccessdisplay{#3}{#2}}%
4798   {\glsglssymbolaccessdisplay{#4}{#2}}%
4799   {\glsglssymbolaccessdisplay{#5}{#2}}{#6}%
4800 }

```

7.4 Acronyms

Use \newacronymhook to modify the key list to set the access text to the long version by default.

```

4801 \renewcommand*{\newacronymhook}{%
4802   \edef\@gls@keylist{\glsshortkey access=\the\glslongtok,%
4803     \the\glskeylisttok}%
4804   \expandafter\glskeylisttok\expandafter{\@gls@keylist}%
4805 }

```

\DefaultNewAcronymDef Modify default style to use access text:

```

4806 \renewcommand*{\DefaultNewAcronymDef}{%
4807   \edef\@do@newglossaryentry{%
4808     \noexpand\newglossaryentry{\the\glslabeltok}%
4809     {%
4810       type=\acronymtype,%
4811       name={\the\glsshorttok},%
4812       description={\the\glslongtok},%
4813       descriptionaccess=\relax,%
4814       text={\the\glsshorttok},%
4815       textaccess={\the\glslongtok},%
4816       access={\noexpand\@glo@textaccess},%
4817       sort={\the\glsshorttok},%
4818       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4819       firstaccess=\relax,%

```

```

4820     first={\noexpand\glsdescriptionaccessdisplay
4821             {\the\glslongtok}{\the\glslabeltok}\space
4822             (\noexpand\glstextaccessdisplay
4823              {\the\glsshorttok}{\the\glslabeltok}}},%
4824     plural={\the\glsshorttok\acrpluralsuffix},%
4825     firstplural={\noexpand\glsdescriptionpluralaccessdisplay
4826                 {\noexpand\@glo@descplural}{\the\glslabeltok}\space
4827                 (\noexpand\glspluralaccessdisplay
4828                  {\noexpand\@glo@plural}{\the\glslabeltok}}},%
4829     firstpluralaccess=\relax,
4830     \the\glskeylisttok
4831 }%
4832 }%
4833 \@do@newglossaryentry
4834 }

```

DescriptionFootnoteNewAcronymDef

```

4835 \renewcommand*{\DescriptionFootnoteNewAcronymDef}{%
4836   \edef\@do@newglossaryentry{%
4837     \noexpand\newglossaryentry{\the\glslabeltok}%
4838     {%
4839       type=\acronymtype,%
4840       name={\noexpand\acronymfont{\the\glsshorttok}},%
4841       sort={\the\glsshorttok},%
4842       text={\the\glsshorttok},%
4843       textaccess={\the\glslongtok},%
4844       access={\noexpand\@glo@textaccess},%
4845       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4846       symbol={\the\glslongtok},%
4847       symbolplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4848       \the\glskeylisttok
4849     }%
4850   }%
4851   \@do@newglossaryentry
4852 }

```

\DescriptionNewAcronymDef

```

4853 \renewcommand*{\DescriptionNewAcronymDef}{%
4854   \edef\@do@newglossaryentry{%
4855     \noexpand\newglossaryentry{\the\glslabeltok}%
4856     {%
4857       type=\acronymtype,%
4858       name={\noexpand
4859             \acronymformat{\the\glsshorttok}{\the\glslongtok}},%
4860       access={\noexpand\@glo@textaccess},%
4861       sort={\the\glsshorttok},%
4862       first={\the\glslongtok},%
4863       firstaccess=\relax,
4864       firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4865       text={\the\glsshorttok},%

```

```

4866     textaccess={\the\glslongtok},%
4867     plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4868     symbol={\noexpand\@glo@text},%
4869     symbolaccess={\noexpand\@glo@textaccess},%
4870     symbolplural={\noexpand\@glo@plural},%
4871     \the\glskeylisttok}%
4872 }%
4873 \@do@newglossaryentry
4874 }

```

\FootnoteNewAcronymDef

```

4875 \renewcommand*\{\FootnoteNewAcronymDef}{%
4876   \edef\@do@newglossaryentry{%
4877     \noexpand\newglossaryentry{\the\glslabeltok}%
4878     {%
4879       type=\acronymtype,%
4880       name={\noexpand\acronymfont{\the\glsshorttok}},%
4881       access={\noexpand\@glo@textaccess},%
4882       sort={\the\glsshorttok},%
4883       text={\the\glsshorttok},%
4884       textaccess={\the\glslongtok},%
4885       plural={\the\glsshorttok\noexpand\acrpluralsuffix},%
4886       description={\the\glslongtok},%
4887       descriptionplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4888       \the\glskeylisttok
4889     }%
4890   }%
4891   \@do@newglossaryentry
4892 }

```

\SmallNewAcronymDef

```

4893 \renewcommand*\{\SmallNewAcronymDef}{%
4894   \edef\@do@newglossaryentry{%
4895     \noexpand\newglossaryentry{\the\glslabeltok}%
4896     {%
4897       type=\acronymtype,%
4898       name={\noexpand\acronymfont{\the\glsshorttok}},%
4899       access={\noexpand\@glo@symbolaccess},%
4900       sort={\the\glsshorttok},%
4901       text={\noexpand\@glo@symbol},%
4902       textaccess={\noexpand\@glo@symbolaccess},%
4903       plural={\noexpand\@glo@symbolplural},%
4904       first={\the\glslongtok},%
4905       firstaccess=\relax,%
4906       firstplural={\the\glslongtok\noexpand\acrpluralsuffix},%
4907       description={\noexpand\@glo@first},%
4908       descriptionplural={\noexpand\@glo@firstplural},%
4909       symbol={\the\glsshorttok},%
4910       symbolaccess={\the\glslongtok},%
4911       symbolplural={\the\glsshorttok\noexpand\acrpluralsuffix},%

```

```

4912     \the\glskeylisttok
4913   }%
4914 }%
4915 \@do@newglossaryentry
4916 }

```

Add means of referencing accessibility support for acronyms:

```

\glsshortaccesskey
4917 \newcommand*{\glsshortaccesskey}{\glsshortkey access}%

\glsshortpluralaccesskey
4918 \newcommand*{\glsshortpluralaccesskey}{\glsshortpluralkey access}%

\glslongaccesskey
4919 \newcommand*{\glslongaccesskey}{\glslongkey access}%

\glslongpluralaccesskey
4920 \newcommand*{\glslongpluralaccesskey}{\glslongpluralkey access}%

```

8 Multi-Lingual Support

Many thanks to everyone who contributed to the translations both via email and on comp.text.tex.

8.1 Babel Captions

Define captions if multi-lingual support is required, but the package is not loaded.

```

4921 \NeedsTeXFormat{LaTeX2e}
4922 \ProvidesPackage{glossaries-babel}[2009/04/16 v1.2 (NLCT)]

English:
4923 \@ifundefined{captionsenglish}{}{%
4924   \addto\captionsenglish{%
4925     \renewcommand*{\glossaryname}{Glossary}%
4926     \renewcommand*{\acronymname}{Acronyms}%
4927     \renewcommand*{\entryname}{Notation}%
4928     \renewcommand*{\descriptionname}{Description}%
4929     \renewcommand*{\symbolname}{Symbol}%
4930     \renewcommand*{\pagelistname}{Page List}%
4931     \renewcommand*{\glssymbolsgroupname}{Symbols}%
4932     \renewcommand*{\glsnumbersgroupname}{Numbers}%
4933   }%
4934 }
4935 \@ifundefined{captionsamerican}{}{%
4936   \addto\captionsamerican{%
4937     \renewcommand*{\glossaryname}{Glossary}%
4938     \renewcommand*{\acronymname}{Acronyms}%

```

```

4939 \renewcommand*{\entryname}{Notation}%
4940 \renewcommand*{\descriptionname}{Description}%
4941 \renewcommand*{\symbolname}{Symbol}%
4942 \renewcommand*{\pagelistname}{Page List}%
4943 \renewcommand*{\glssymbolsgroupname}{Symbols}%
4944 \renewcommand*{\glsnumbersgroupname}{Numbers}%
4945 }%
4946 }
4947 \@ifundefined{captionsaustralian}{}{%
4948 \addto\captionsaustralian{%
4949 \renewcommand*{\glossaryname}{Glossary}%
4950 \renewcommand*{\acronymname}{Acronyms}%
4951 \renewcommand*{\entryname}{Notation}%
4952 \renewcommand*{\descriptionname}{Description}%
4953 \renewcommand*{\symbolname}{Symbol}%
4954 \renewcommand*{\pagelistname}{Page List}%
4955 \renewcommand*{\glssymbolsgroupname}{Symbols}%
4956 \renewcommand*{\glsnumbersgroupname}{Numbers}%
4957 }%
4958 }
4959 \@ifundefined{captionsbritish}{}{%
4960 \addto\captionsbritish{%
4961 \renewcommand*{\glossaryname}{Glossary}%
4962 \renewcommand*{\acronymname}{Acronyms}%
4963 \renewcommand*{\entryname}{Notation}%
4964 \renewcommand*{\descriptionname}{Description}%
4965 \renewcommand*{\symbolname}{Symbol}%
4966 \renewcommand*{\pagelistname}{Page List}%
4967 \renewcommand*{\glssymbolsgroupname}{Symbols}%
4968 \renewcommand*{\glsnumbersgroupname}{Numbers}%
4969 }}%
4970 \@ifundefined{captionscanadian}{}{%
4971 \addto\captionscanadian{%
4972 \renewcommand*{\glossaryname}{Glossary}%
4973 \renewcommand*{\acronymname}{Acronyms}%
4974 \renewcommand*{\entryname}{Notation}%
4975 \renewcommand*{\descriptionname}{Description}%
4976 \renewcommand*{\symbolname}{Symbol}%
4977 \renewcommand*{\pagelistname}{Page List}%
4978 \renewcommand*{\glssymbolsgroupname}{Symbols}%
4979 \renewcommand*{\glsnumbersgroupname}{Numbers}%
4980 }%
4981 }
4982 \@ifundefined{captionsnewzealand}{}{%
4983 \addto\captionsnewzealand{%
4984 \renewcommand*{\glossaryname}{Glossary}%
4985 \renewcommand*{\acronymname}{Acronyms}%
4986 \renewcommand*{\entryname}{Notation}%
4987 \renewcommand*{\descriptionname}{Description}%
4988 \renewcommand*{\symbolname}{Symbol}%

```

```

4989 \renewcommand*{\pagelistname}{Page List}%
4990 \renewcommand*{\glssymbolsgroupname}{Symbols}%
4991 \renewcommand*{\glsnumbersgroupname}{Numbers}%
4992 }%
4993 }
4994 \@ifundefined{captionsUKenglish}{}{%
4995 \addto\captionsUKenglish{%
4996 \renewcommand*{\glossaryname}{Glossary}%
4997 \renewcommand*{\acronymname}{Acronyms}%
4998 \renewcommand*{\entryname}{Notation}%
4999 \renewcommand*{\descriptionname}{Description}%
5000 \renewcommand*{\symbolname}{Symbol}%
5001 \renewcommand*{\pagelistname}{Page List}%
5002 \renewcommand*{\glssymbolsgroupname}{Symbols}%
5003 \renewcommand*{\glsnumbersgroupname}{Numbers}%
5004 }%
5005 }
5006 \@ifundefined{captionsUSenglish}{}{%
5007 \addto\captionsUSenglish{%
5008 \renewcommand*{\glossaryname}{Glossary}%
5009 \renewcommand*{\acronymname}{Acronyms}%
5010 \renewcommand*{\entryname}{Notation}%
5011 \renewcommand*{\descriptionname}{Description}%
5012 \renewcommand*{\symbolname}{Symbol}%
5013 \renewcommand*{\pagelistname}{Page List}%
5014 \renewcommand*{\glssymbolsgroupname}{Symbols}%
5015 \renewcommand*{\glsnumbersgroupname}{Numbers}%
5016 }%
5017 }

```

German (quite a few variations were suggested for German; I settled on the following):

```

5018 \@ifundefined{captionsgerman}{}{%
5019 \addto\captionsgerman{%
5020 \renewcommand*{\glossaryname}{Glossar}%
5021 \renewcommand*{\acronymname}{Akronyme}%
5022 \renewcommand*{\entryname}{Bezeichnung}%
5023 \renewcommand*{\descriptionname}{Beschreibung}%
5024 \renewcommand*{\symbolname}{Symbol}%
5025 \renewcommand*{\pagelistname}{Seiten}%
5026 \renewcommand*{\glssymbolsgroupname}{Symbole}%
5027 \renewcommand*{\glsnumbersgroupname}{Zahlen}}
5028 }

```

ngerman is identical to German:

```

5029 \@ifundefined{captionsgerman}{}{%
5030 \addto\captionsgerman{%
5031 \renewcommand*{\glossaryname}{Glossar}%
5032 \renewcommand*{\acronymname}{Akronyme}%
5033 \renewcommand*{\entryname}{Bezeichnung}%
5034 \renewcommand*{\descriptionname}{Beschreibung}%

```

```

5035 \renewcommand*{\symbolname}{Symbol}%
5036 \renewcommand*{\pagelistname}{Seiten}%
5037 \renewcommand*{\glssymbolsgroupname}{Symbole}%
5038 \renewcommand*{\glsnumbersgroupname}{Zahlen}%
5039 }

```

Italian:

```

5040 \@ifundefined{captionsitalian}{}{%
5041 \addto\captionsitalian{%
5042 \renewcommand*{\glossaryname}{Glossario}%
5043 \renewcommand*{\acronymname}{Acronimi}%
5044 \renewcommand*{\entryname}{Nomenclatura}%
5045 \renewcommand*{\descriptionname}{Descrizione}%
5046 \renewcommand*{\symbolname}{Simbolo}%
5047 \renewcommand*{\pagelistname}{Elenco delle pagine}%
5048 \renewcommand*{\glssymbolsgroupname}{Simboli}%
5049 \renewcommand*{\glsnumbersgroupname}{Numeri}}
5050 }

```

Dutch:

```

5051 \@ifundefined{captionsdutch}{}{%
5052 \addto\captionsdutch{%
5053 \renewcommand*{\glossaryname}{Woordenlijst}%
5054 \renewcommand*{\acronymname}{Acroniemen}%
5055 \renewcommand*{\entryname}{Benaming}%
5056 \renewcommand*{\descriptionname}{Beschrijving}%
5057 \renewcommand*{\symbolname}{Symbool}%
5058 \renewcommand*{\pagelistname}{Pagina's}%
5059 \renewcommand*{\glssymbolsgroupname}{Symbolen}%
5060 \renewcommand*{\glsnumbersgroupname}{Cijfers}}
5061 }

```

Spanish:

```

5062 \@ifundefined{captionsspanish}{}{%
5063 \addto\captionsspanish{%
5064 \renewcommand*{\glossaryname}{Glosario}%
5065 \renewcommand*{\acronymname}{Siglas}%
5066 \renewcommand*{\entryname}{Entrada}%
5067 \renewcommand*{\descriptionname}{Descripci\'on}%
5068 \renewcommand*{\symbolname}{S\'mbolo}%
5069 \renewcommand*{\pagelistname}{Lista de p\'aginas}%
5070 \renewcommand*{\glssymbolsgroupname}{S\'mbolos}%
5071 \renewcommand*{\glsnumbersgroupname}{N\'umeros}}
5072 }

```

French:

```

5073 \@ifundefined{captionsfrench}{}{%
5074 \addto\captionsfrench{%
5075 \renewcommand*{\glossaryname}{Glossaire}%
5076 \renewcommand*{\acronymname}{Acronymes}%
5077 \renewcommand*{\entryname}{Termes}%

```



```

5078 \renewcommand*{\descriptionname}{Description}%
5079 \renewcommand*{\symbolname}{Symbole}%
5080 \renewcommand*{\pagelistname}{Pages}%
5081 \renewcommand*{\glssymbolsgroupname}{Symboles}%
5082 \renewcommand*{\glsnumbersgroupname}{Nombres}}
5083 }
5084 \@ifundefined{captionsfrenchb}{}{%
5085 \addto\captionsfrenchb{%
5086 \renewcommand*{\glossaryname}{Glossaire}%
5087 \renewcommand*{\acronymname}{Acronymes}%
5088 \renewcommand*{\entryname}{Terme}%
5089 \renewcommand*{\descriptionname}{Description}%
5090 \renewcommand*{\symbolname}{Symbole}%
5091 \renewcommand*{\pagelistname}{Pages}%
5092 \renewcommand*{\glssymbolsgroupname}{Symboles}%
5093 \renewcommand*{\glsnumbersgroupname}{Nombres}}
5094 }
5095 \@ifundefined{captionsfrançais}{}{%
5096 \addto\captionsfrançais{%
5097 \renewcommand*{\glossaryname}{Glossaire}%
5098 \renewcommand*{\acronymname}{Acronymes}%
5099 \renewcommand*{\entryname}{Terme}%
5100 \renewcommand*{\descriptionname}{Description}%
5101 \renewcommand*{\symbolname}{Symbole}%
5102 \renewcommand*{\pagelistname}{Pages}%
5103 \renewcommand*{\glssymbolsgroupname}{Symboles}%
5104 \renewcommand*{\glsnumbersgroupname}{Nombres}}
5105 }

```

Danish:

```

5106 \@ifundefined{captionsdanish}{}{%
5107 \addto\captionsdanish{%
5108 \renewcommand*{\glossaryname}{Ordliste}%
5109 \renewcommand*{\acronymname}{Akronymer}%
5110 \renewcommand*{\entryname}{Symbolforklaring}%
5111 \renewcommand*{\descriptionname}{Beskrivelse}%
5112 \renewcommand*{\symbolname}{Symbol}%
5113 \renewcommand*{\pagelistname}{Side}%
5114 \renewcommand*{\glssymbolsgroupname}{Symboler}%
5115 \renewcommand*{\glsnumbersgroupname}{Tal}}
5116 }

```

Irish:

```

5117 \@ifundefined{captionsirish}{}{%
5118 \addto\captionsirish{%
5119 \renewcommand*{\glossaryname}{Glúais}%
5120 \renewcommand*{\acronymname}{Acrainmneacha}%

```

wasn't sure whether to go for Nóta (Note), Ciall ('Meaning', 'sense') or Brí ('Meaning'). In the end I chose Ciall.

```

5121 \renewcommand*{\entryname}{Ciall}%

```

```

5122 \renewcommand*{\descriptionname}{Túairisc}%
    Again, not sure whether to use Comhartha/Comharthaí or Siombail/Siombaile,
    so have chosen the former.
5123 \renewcommand*{\symbolname}{Comhartha}%
5124 \renewcommand*{\glssymbolsgroupname}{Comhartha\'}{\i}%
5125 \renewcommand*{\pagelistname}{Leathanaigh}%
5126 \renewcommand*{\glsnumbersgroupname}{Uimhreacha}}
5127 }

    Hungarian:
5128 \@ifundefined{captionsmagyar}{}{%
5129 \addto\captionsmagyar{%
5130 \renewcommand*{\glossaryname}{Sz\'ojegyz\'ek}%
5131 \renewcommand*{\acronymname}{Bet\H uszavak}%
5132 \renewcommand*{\entryname}{Kifejez\'es}%
5133 \renewcommand*{\descriptionname}{Magyar\'azat}%
5134 \renewcommand*{\symbolname}{Jel\'ol\'es}%
5135 \renewcommand*{\pagelistname}{Oldalsz\'am}%
5136 \renewcommand*{\glssymbolsgroupname}{Jelek}%
5137 \renewcommand*{\glsnumbersgroupname}{Sz\'amjegyek}%
5138 }
5139 }

5140 \@ifundefined{captionshungarian}{}{%
5141 \addto\captionshungarian{%
5142 \renewcommand*{\glossaryname}{Sz\'ojegyz\'ek}%
5143 \renewcommand*{\acronymname}{Bet\H uszavak}%
5144 \renewcommand*{\entryname}{Kifejez\'es}%
5145 \renewcommand*{\descriptionname}{Magyar\'azat}%
5146 \renewcommand*{\symbolname}{Jel\'ol\'es}%
5147 \renewcommand*{\pagelistname}{Oldalsz\'am}%
5148 \renewcommand*{\glssymbolsgroupname}{Jelek}%
5149 \renewcommand*{\glsnumbersgroupname}{Sz\'amjegyek}%
5150 }
5151 }

    Polish
5152 \@ifundefined{captionspolish}{}{%
5153 \addto\captionspolish{%
5154 \renewcommand*{\glossaryname}{S\lownik termin\'ow}%
5155 \renewcommand*{\acronymname}{Skr\'ot}%
5156 \renewcommand*{\entryname}{Termin}%
5157 \renewcommand*{\descriptionname}{Opis}%
5158 \renewcommand*{\symbolname}{Symbol}%
5159 \renewcommand*{\pagelistname}{Strony}%
5160 \renewcommand*{\glssymbolsgroupname}{Symbole}%
5161 \renewcommand*{\glsnumbersgroupname}{Liczby}}
5162 }

    Brazilian
5163 \@ifundefined{captionsbrazil}{}{%

```

```

5164 \addto\captionsbrazil{%
5165   \renewcommand*{\glossaryname}{Gloss\'ario}%
5166   \renewcommand*{\acronymname}{Siglas}%
5167   \renewcommand*{\entryname}{Nota\ c\~ao}%
5168   \renewcommand*{\descriptionname}{Descri\ c\~ao}%
5169   \renewcommand*{\symbolname}{S\'imbolo}%
5170   \renewcommand*{\pagelistname}{Lista de P\'aginas}%
5171   \renewcommand*{\glssymbolsgroupname}{S\'imbolos}%
5172   \renewcommand*{\glsnumbersgroupname}{N\'umeros}%
5173 }%
5174 }

```

8.2 Polyglossia Captions

```

5175 \NeedsTeXFormat{LaTeX2e}
5176 \ProvidesPackage{glossaries-polyglossia}[2009/11/09 v1.0 (NLCT)]

```

English:

```

5177 \@ifundefined{captionsenglish}{-}{%
5178   \expandafter\toks@\expandafter{\captionsenglish
5179     \renewcommand*{\glossaryname}{\textenglish{Glossary}}%
5180     \renewcommand*{\acronymname}{\textenglish{Acronyms}}%
5181     \renewcommand*{\entryname}{\textenglish{Notation}}%
5182     \renewcommand*{\descriptionname}{\textenglish{Description}}%
5183     \renewcommand*{\symbolname}{\textenglish{Symbol}}%
5184     \renewcommand*{\pagelistname}{\textenglish{Page List}}%
5185     \renewcommand*{\glssymbolsgroupname}{\textenglish{Symbols}}%
5186     \renewcommand*{\glsnumbersgroupname}{\textenglish{Numbers}}%
5187   }%
5188   \edef\captionsenglish{\the\toks@}%
5189 }

```

German:

```

5190 \@ifundefined{captionsgerman}{-}{%
5191   \expandafter\toks@\expandafter{\captionsgerman
5192     \renewcommand*{\glossaryname}{\textgerman{Glossar}}%
5193     \renewcommand*{\acronymname}{\textgerman{Akronyme}}%
5194     \renewcommand*{\entryname}{\textgerman{Bezeichnung}}%
5195     \renewcommand*{\descriptionname}{\textgerman{Beschreibung}}%
5196     \renewcommand*{\symbolname}{\textgerman{Symbol}}%
5197     \renewcommand*{\pagelistname}{\textgerman{Seiten}}%
5198     \renewcommand*{\glssymbolsgroupname}{\textgerman{Symbole}}%
5199     \renewcommand*{\glsnumbersgroupname}{\textgerman{Zahlen}}%
5200   }%
5201   \edef\captionsgerman{\the\toks@}%
5202 }

```

Italian:

```

5203 \@ifundefined{captionsitalian}{-}{%
5204   \expandafter\toks@\expandafter{\captionsitalian
5205     \renewcommand*{\glossaryname}{\textitalian{Glossario}}%

```

```

5206 \renewcommand*{\acronymname}{\textitalian{Acronimi}}%
5207 \renewcommand*{\entryname}{\textitalian{Nomenclatura}}%
5208 \renewcommand*{\descriptionname}{\textitalian{Descrizione}}%
5209 \renewcommand*{\symbolname}{\textitalian{Simbolo}}%
5210 \renewcommand*{\pagelistname}{\textitalian{Elenco delle pagine}}%
5211 \renewcommand*{\glssymbolsgroupname}{\textitalian{Simboli}}%
5212 \renewcommand*{\glslnumbersgroupname}{\textitalian{Numeri}}%
5213 }%
5214 \edef\captionssitalian{\the\toks@}%
5215 }

```

Dutch:

```

5216 \@ifundefined{captionsdutch}{\{%
5217 \expandafter\toks@\expandafter{\captionsdutch
5218 \renewcommand*{\glossaryname}{\textdutch{Woordenlijst}}%
5219 \renewcommand*{\acronymname}{\textdutch{Acroniemen}}%
5220 \renewcommand*{\entryname}{\textdutch{Benaming}}%
5221 \renewcommand*{\descriptionname}{\textdutch{Beschrijving}}%
5222 \renewcommand*{\symbolname}{\textdutch{Symbool}}%
5223 \renewcommand*{\pagelistname}{\textdutch{Pagina's}}%
5224 \renewcommand*{\glssymbolsgroupname}{\textdutch{Symbolen}}%
5225 \renewcommand*{\glslnumbersgroupname}{\textdutch{Cijfers}}%
5226 }%
5227 \edef\captionsdutch{\the\toks@}%
5228 }

```

Spanish:

```

5229 \@ifundefined{captionsspanish}{\{%
5230 \expandafter\toks@\expandafter{\captionsspanish
5231 \renewcommand*{\glossaryname}{\textspanish{Glosario}}%
5232 \renewcommand*{\acronymname}{\textspanish{Siglas}}%
5233 \renewcommand*{\entryname}{\textspanish{Entrada}}%
5234 \renewcommand*{\descriptionname}{\textspanish{Descripci'on}}%
5235 \renewcommand*{\symbolname}{\textspanish{S'\i mbolo}}%
5236 \renewcommand*{\pagelistname}{\textspanish{Lista de p'aginas}}%
5237 \renewcommand*{\glssymbolsgroupname}{\textspanish{S'\i mbolos}}%
5238 \renewcommand*{\glslnumbersgroupname}{\textspanish{N'umeros}}%
5239 }%
5240 \edef\captionsspanish{\the\toks@}%
5241 }

```

French:

```

5242 \@ifundefined{captionsfrench}{\{%
5243 \expandafter\toks@\expandafter{\captionsfrench
5244 \renewcommand*{\glossaryname}{\textfrench{Glossaire}}%
5245 \renewcommand*{\acronymname}{\textfrench{Acronymes}}%
5246 \renewcommand*{\entryname}{\textfrench{Terme}}%
5247 \renewcommand*{\descriptionname}{\textfrench{Description}}%
5248 \renewcommand*{\symbolname}{\textfrench{Symbole}}%
5249 \renewcommand*{\pagelistname}{\textfrench{Pages}}%
5250 \renewcommand*{\glssymbolsgroupname}{\textfrench{Symboles}}%
5251 \renewcommand*{\glslnumbersgroupname}{\textfrench{Nombres}}%

```

```

5252 }%
5253 \edef\captionfrench{\the\toks@}%
5254 }

Danish:
5255 \@ifundefined{captionsdanish}{}{%
5256   \expandafter\toks@\expandafter{\captionsdanish
5257     \renewcommand*\glossaryname{\textdanish{Ordliste}}%
5258     \renewcommand*\acronymname{\textdanish{Akronymer}}%
5259     \renewcommand*\entryname{\textdanish{Symbolforklaring}}%
5260     \renewcommand*\descriptionname{\textdanish{Beskrivelse}}%
5261     \renewcommand*\symbolname{\textdanish{Symbol}}%
5262     \renewcommand*\pagelistname{\textdanish{Side}}%
5263     \renewcommand*\glssymbolsgroupname{\textdanish{Symboler}}%
5264     \renewcommand*\glslnumbersgroupname{\textdanish{Tal}}%
5265   }%
5266   \edef\captionsdanish{\the\toks@}%
5267 }

Irish:
5268 \@ifundefined{captionsirish}{}{%
5269   \expandafter\toks@\expandafter{\captionsirish
5270     \renewcommand*\glossaryname{\textirish{Gluais}}%
5271     \renewcommand*\acronymname{\textirish{Acrainmneacha}}%
5272     \renewcommand*\entryname{\textirish{Ciall}}%
5273     \renewcommand*\descriptionname{\textirish{Tuaireasc}}%
5274     \renewcommand*\symbolname{\textirish{Comhartha}}%
5275     \renewcommand*\glssymbolsgroupname{\textirish{Comhartha\'{i}}}%
5276     \renewcommand*\pagelistname{\textirish{Leathanaigh}}%
5277     \renewcommand*\glslnumbersgroupname{\textirish{Uimhreacha}}%
5278   }%
5279   \edef\captionsirish{\the\toks@}%
5280 }

Hungarian:
5281 \@ifundefined{captionsmagyar}{}{%
5282   \expandafter\toks@\expandafter{\captionsmagyar
5283     \renewcommand*\glossaryname{\textmagyar{Sz\'ojegyz\'ek}}%
5284     \renewcommand*\acronymname{\textmagyar{Bet\H uszavak}}%
5285     \renewcommand*\entryname{\textmagyar{Kifejez\'es}}%
5286     \renewcommand*\descriptionname{\textmagyar{Magyar\'azat}}%
5287     \renewcommand*\symbolname{\textmagyar{Jel\'ol\'es}}%
5288     \renewcommand*\pagelistname{\textmagyar{Oldalsz\'am}}%
5289     \renewcommand*\glssymbolsgroupname{\textmagyar{Jelek}}%
5290     \renewcommand*\glslnumbersgroupname{\textmagyar{Sz\'amjegyek}}%
5291   }%
5292   \edef\captionsmagyar{\the\toks@}%
5293 }

Polish
5294 \@ifundefined{captionspolish}{}{%
5295   \expandafter\toks@\expandafter{\captionspolish

```

```

5296 \renewcommand*{\glossaryname}{\textpolish{S{\l}ownik termin\ow}}%
5297 \renewcommand*{\acronymname}{\textpolish{Skr\ot}}%
5298 \renewcommand*{\entryname}{\textpolish{Termin}}%
5299 \renewcommand*{\descriptionname}{\textpolish{Opis}}%
5300 \renewcommand*{\symbolname}{\textpolish{Symbol}}%
5301 \renewcommand*{\pagelistname}{\textpolish{Strony}}%
5302 \renewcommand*{\glssymbolsgroupname}{\textpolish{Symbole}}%
5303 \renewcommand*{\glsnumbersgroupname}{\textpolish{Liczby}}%
5304 }%
5305 \edef\captionspolish{\the\toks@}%
5306 }

Portugues
5307 \@ifundefined{captionsportuges}{}%
5308 \expandafter\toks@\expandafter{\captionspolish
5309 \renewcommand*{\glossaryname}{\textportuges{Gloss\ario}}%
5310 \renewcommand*{\acronymname}{\textportuges{Siglas}}%
5311 \renewcommand*{\entryname}{\textportuges{Nota\c c\~ao}}%
5312 \renewcommand*{\descriptionname}{\textportuges{Descri\c c\~ao}}%
5313 \renewcommand*{\symbolname}{\textportuges{S\imbolo}}%
5314 \renewcommand*{\pagelistname}{\textportuges{Lista de P\aginas}}%
5315 \renewcommand*{\glssymbolsgroupname}{\textportuges{S\imbolos}}%
5316 \renewcommand*{\glsnumbersgroupname}{\textportuges{N\umeros}}%
5317 }%
5318 \edef\captionsportuges{\the\toks@}%
5319 }

```

8.3 Brazilian Dictionary

This is a dictionary file provided by Thiago de Melo for use with the package.

```

5320 \ProvidesDictionary{glossaries-dictionary}{Brazilian}

Provide Brazilian translations:
5321 \providetranslation{Glossary}{Gloss\ario}
5322 \providetranslation{Acronyms}{Siglas}
5323 \providetranslation{Notation (glossaries)}{Nota\c c\~ao}
5324 \providetranslation{Description (glossaries)}{Descri\c c\~ao}
5325 \providetranslation{Symbol (glossaries)}{S\imbolo}
5326 \providetranslation{Page List (glossaries)}{Lista de P\aginas}
5327 \providetranslation{Symbols (glossaries)}{S\imbolos}
5328 \providetranslation{Numbers (glossaries)}{N\umeros}

```

8.4 Danish Dictionary

This is a dictionary file provided for use with the package.

```

5329 \ProvidesDictionary{glossaries-dictionary}{Danish}

Provide Danish translations:
5330 \providetranslation{Glossary}{Ordliste}
5331 \providetranslation{Acronyms}{Akronymer}
5332 \providetranslation{Notation (glossaries)}{Symbolforklaring}

```

```

5333 \providetranslation{Description (glossaries)}{Beskrivelse}
5334 \providetranslation{Symbol (glossaries)}{Symbol}
5335 \providetranslation{Page List (glossaries)}{Side}
5336 \providetranslation{Symbols (glossaries)}{Symboler}
5337 \providetranslation{Numbers (glossaries)}{Tal}

```

8.5 Dutch Dictionary

This is a dictionary file provided for use with the package.

```
5338 \ProvidesDictionary{glossaries-dictionary}{Dutch}
```

Provide Dutch translations:

```

5339 \providetranslation{Glossary}{Woordenlijst}
5340 \providetranslation{Acronyms}{Acroniemen}
5341 \providetranslation{Notation (glossaries)}{Benaming}
5342 \providetranslation{Description (glossaries)}{Beschrijving}
5343 \providetranslation{Symbol (glossaries)}{Symbool}
5344 \providetranslation{Page List (glossaries)}{Pagina's}
5345 \providetranslation{Symbols (glossaries)}{Symbolen}
5346 \providetranslation{Numbers (glossaries)}{Cijfers}

```

8.6 English Dictionary

This is a dictionary file provided for use with the package.

```
5347 \ProvidesDictionary{glossaries-dictionary}{English}
```

Provide English translations:

```

5348 \providetranslation{Glossary}{Glossary}
5349 \providetranslation{Acronyms}{Acronyms}
5350 \providetranslation{Notation (glossaries)}{Notation}
5351 \providetranslation{Description (glossaries)}{Description}
5352 \providetranslation{Symbol (glossaries)}{Symbol}
5353 \providetranslation{Page List (glossaries)}{Page List}
5354 \providetranslation{Symbols (glossaries)}{Symbols}
5355 \providetranslation{Numbers (glossaries)}{Numbers}

```

8.7 French Dictionary

This is a dictionary file provided for use with the package.

```
5356 \ProvidesDictionary{glossaries-dictionary}{French}
```

Provide French translations:

```

5357 \providetranslation{Glossary}{Glossaire}
5358 \providetranslation{Acronyms}{Acronymes}
5359 \providetranslation{Notation (glossaries)}{Termes}
5360 \providetranslation{Description (glossaries)}{Description}
5361 \providetranslation{Symbol (glossaries)}{Symbole}
5362 \providetranslation{Page List (glossaries)}{Pages}
5363 \providetranslation{Symbols (glossaries)}{Symboles}
5364 \providetranslation{Numbers (glossaries)}{Nombres}

```

8.8 German Dictionary

This is a dictionary file provided for use with the package.

```
5365 \ProvidesDictionary{glossaries-dictionary}{German}
```

Provide German translations (quite a few variations were suggested for German; I settled on the following):

```
5366 \providetranslation{Glossary}{Glossar}
5367 \providetranslation{Acronyms}{Akronyme}
5368 \providetranslation{Notation (glossaries)}{Bezeichnung}
5369 \providetranslation{Description (glossaries)}{Beschreibung}
5370 \providetranslation{Symbol (glossaries)}{Symbol}
5371 \providetranslation{Page List (glossaries)}{Seiten}
5372 \providetranslation{Symbols (glossaries)}{Symbole}
5373 \providetranslation{Numbers (glossaries)}{Zahlen}
```

8.9 Irish Dictionary

This is a dictionary file provided for use with the package.

```
5374 \ProvidesDictionary{glossaries-dictionary}{Irish}
```

Provide Irish translations:

```
5375 \providetranslation{Glossary}{Gluais}
5376 \providetranslation{Acronyms}{Acrainmneacha}
5377 \providetranslation{Notation (glossaries)}{Ciall}
5378 \providetranslation{Description (glossaries)}{Tuairisc}
5379 \providetranslation{Symbol (glossaries)}{Comhartha}
5380 \providetranslation{Page List (glossaries)}{Leathanaigh}
5381 \providetranslation{Symbols (glossaries)}{Comhartha\'}{\i}
5382 \providetranslation{Numbers (glossaries)}{Uimhreacha}
```

8.10 Italian Dictionary

This is a dictionary file provided for use with the package.

```
5383 \ProvidesDictionary{glossaries-dictionary}{Italian}
```

Provide Italian translations:

```
5384 \providetranslation{Glossary}{Glossario}
5385 \providetranslation{Acronyms}{Acronimi}
5386 \providetranslation{Notation (glossaries)}{Nomenclatura}
5387 \providetranslation{Description (glossaries)}{Descrizione}
5388 \providetranslation{Symbol (glossaries)}{Simbolo}
5389 \providetranslation{Page List (glossaries)}{Elenco delle pagine}
5390 \providetranslation{Symbols (glossaries)}{Simboli}
5391 \providetranslation{Numbers (glossaries)}{Numeri}
```

8.11 Magyar Dictionary

This is a dictionary file provided for use with the package.

```
5392 \ProvidesDictionary{glossaries-dictionary}{Magyar}
```


Provide translations:

```
5393 \providetranslation{Glossary}{Sz\'ojegyz\'ek}
5394 \providetranslation{Acronyms}{Bet\H uszavak}
5395 \providetranslation{Notation (glossaries)}{Kifejez\'es}
5396 \providetranslation{Description (glossaries)}{Magyar\'azat}
5397 \providetranslation{Symbol (glossaries)}{Jel\'ol\'es}
5398 \providetranslation{Page List (glossaries)}{Oldalsz\'am}
5399 \providetranslation{Symbols (glossaries)}{Jelek}
5400 \providetranslation{Numbers (glossaries)}{Sz\'amjegyek}
```

8.12 Polish Dictionary

This is a dictionary file provided for use with the package.

```
5401 \ProvidesDictionary{glossaries-dictionary}{Polish}
```

Provide Polish translations:

```
5402 \providetranslation{Glossary}{S{\l}ownik termin\'ow}
5403 \providetranslation{Acronyms}{Skr\'ot}
5404 \providetranslation{Notation (glossaries)}{Termin}
5405 \providetranslation{Description (glossaries)}{Opis}
5406 \providetranslation{Symbol (glossaries)}{Symbol}
5407 \providetranslation{Page List (glossaries)}{Strony}
5408 \providetranslation{Symbols (glossaries)}{Symbole}
5409 \providetranslation{Numbers (glossaries)}{Liczby}
```

8.13 Serbian Dictionary

This dictionary was provided by Zoran Filipovic.

```
5410 \ProvidesDictionary{glossaries-dictionary}{Serbian}
5411 \providetranslation{Glossary}{Mali re\vnik}
5412 \providetranslation{Acronyms}{Skra\'cenice}
5413 \providetranslation{Notation (glossaries)}{Oznaka}
5414 \providetranslation{Description (glossaries)}{Opis}
5415 \providetranslation{Symbol (glossaries)}{Simbol}
5416 \providetranslation{Page List (glossaries)}{Stranica}
5417 \providetranslation{Symbols (glossaries)}{Simboli}
5418 \providetranslation{Numbers (glossaries)}{Brojevi}
```

8.14 Spanish Dictionary

This is a dictionary file provided for use with the package.

```
5419 \ProvidesDictionary{glossaries-dictionary}{Spanish}
```

Provide Spanish translations:

```
5420 \providetranslation{Glossary}{Glosario}
5421 \providetranslation{Acronyms}{Siglas}
5422 \providetranslation{Notation (glossaries)}{Entrada}
5423 \providetranslation{Description (glossaries)}{Descripci\'on}
5424 \providetranslation{Symbol (glossaries)}{S\'imbolo}
```

5425 \providetranslation{Page List (glossaries)}{Lista de p\'aginas}
5426 \providetranslation{Symbols (glossaries)}{S\'{\i}mbolos}
5427 \providetranslation{Numbers (glossaries)}{N\'umeros}

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols	
\@glossarysec	<u>89</u>
\@glossaryseclabel	<u>90</u>
\@glossarysecstar	<u>89</u>
\GLS@	<u>141</u> , <u>261</u>
\GLSpl	<u>145</u>
\GLSpl@	<u>264</u>
\Gls@	<u>140</u> , <u>260</u>
\Glspl@	<u>143</u> , <u>263</u>
\@addtoacronymlists	<u>92</u>
\@delimN	<u>192</u>
\@delimR	<u>192</u>
\@disable@onlypremakeg	<u>98</u>
\@disable@premakecs	<u>98</u>
\@do@seeglossary	<u>183</u>
\@do@wrglossary	<u>183</u>
\@glo@storeentry	<u>124</u>
\@glo@types	<u>114</u>
\@glossary	<u>182</u>
\@glossary@default@style	<u>90</u>
\@glossaryentryfield	<u>124</u> , <u>265</u>
\@glossarysection	<u>105</u>
\@glossarysubentryfield	<u>124</u> , <u>265</u>
\@gls	<u>139</u>
\@gls@	<u>139</u> , <u>259</u>
\@gls@@link	<u>130</u>
\@gls@checkactual	<u>136</u>
\@gls@checkbar	<u>135</u>
\@gls@checkescactual	<u>133</u>
\@gls@checkescbar	<u>134</u>
\@gls@checkesclevel	<u>134</u>
\@gls@checkescquote	<u>133</u>
\@gls@checklevel	<u>135</u>
\@gls@checkmkidxchars	<u>132</u>
\@gls@checkquote	<u>132</u>
\@gls@codepage	<u>20</u> , <u>112</u>
\@gls@escbsdq	<u>131</u>
\@gls@fixbraces	<u>184</u>
\@gls@getcounter	<u>116</u>
\@gls@hypergroup	<u>220</u>
\@gls@link	<u>130</u>
\@gls@loadlist	<u>91</u>
\@gls@loadlong	<u>91</u>
\@gls@loadsuper	<u>91</u>
\@gls@loadtree	<u>91</u>
\@gls@noaccess	<u>255</u>
\@gls@onlypremakeg	<u>98</u>
\@gls@pl@	<u>262</u>
\@gls@renewglossary	<u>182</u>
\@gls@sanitizedesc	<u>94</u>
\@gls@sanitizename	<u>94</u>
\@gls@sanitizesymbol	<u>94</u>
\@gls@setcounter	<u>116</u>
\@gls@tmpb	<u>132</u>
\@gls@toc	<u>106</u>
\@gls@updatechecked	<u>132</u>
\@gls@xdycheckbackslash	<u>137</u>
\@gls@xdycheckquote	<u>136</u>
\@gls@Alphacompositor	<u>103</u> , <u>176</u>
\@gls@acronymlists	<u>92</u>
\@gls@defaultplural	<u>119</u>
\@gls@defaultsort	<u>119</u>
\@gls@disp	<u>146</u>
\@gls@firstletter	<u>174</u>
\@gls@hypernumber	<u>192</u>
\@gls@link	<u>138</u>
\@gls@minrange	<u>174</u>
\@gls@noname	<u>119</u>
\@gls@nonextpages	<u>188</u>
\@gls@order	<u>20</u> , <u>96</u>
\@gls@pl@	<u>142</u>
\@gls@target	<u>138</u>
\@gls@widestname	<u>251</u>
\@istfilename	<u>20</u> , <u>102</u>
\@makeglossary	<u>180</u>
\@newglossary	<u>19</u> , <u>116</u>
\@newglossaryentryposthook	<u>124</u>
\@newglossaryentryprehook	<u>124</u>
\@nopostdesc	<u>101</u>
\@onlypremakeg	<u>98</u>
\@p@glossarysection	<u>106</u>
\@set@glo@numformat	<u>131</u>
\@sgls	<u>139</u> , <u>145</u>
\@sgls@link	<u>130</u>
\@wrglossary	<u>182</u>
\@xdy@main@language	<u>97</u>

\@xdyattributes	107	\acrshort	60, 66, 67, 195
\@xdylanguage	20, 111	\ACRshortpl	196
\@xdylettergroups	112	\Acrshortpl	67, 196
\@xdylocationclassorder	109	\acrshortpl	67, 195
\@xdylocref	107	\Acs	67, 215
\@xdyrequiredstyles	109	\acs	67, 215
\@xdysortrules	109	\Acsp	67, 215
\@xdyuseralphabets	108	\acsp	67, 215
\@xdyuserlocationdefs	108	\addglossarytocaptions	100
\@xdyuserlocationnames	108	\addto	100
A			
\Ac	67, 216	altlist (style)	222
\ac	67, 216	altlistgroup (style)	223
access (key)	254	altlisthypergroup (style)	223
accsupp package	87, 253	altlong4col (style)	228
\accsuppglossaryentryfield	266	altlong4colborder (style)	229
\accsuppglossarysubentryfield	266	altlong4colheader (style)	228
\Acf	67, 216	altlong4colheaderborder (style)	229
\acf	67, 216	altlongragged4col (style)	233
\Acfp	67, 216	altlongragged4colborder (style)	233
\acfp	67, 216	altlongragged4colheader (style)	233
\Acl	67, 216	altlongragged4colheaderborder	
\acl	67, 215	(style)	234
\Aclp	67, 216	\altnewglossary	115
\aclp	67, 216	altsuper4col (style)	239
\Acp	67, 216	altsuper4colborder (style)	240
\acp	67, 216	altsuper4colheader (style)	239
\ACRfull	66, 196	altsuper4colheaderborder (style)	240
\Acrfull	66, 67, 196	altsuperragged4col (style)	244
\acrfull	66, 67, 196	altsuperragged4colborder (style)	245
\ACRfullpl	197	altsuperragged4colheader (style)	245
\Acrfullpl	67, 197	altsuperragged4colheaderborder	
\acrfullpl	67, 197	(style)	245
\ACRlong	66, 196	alttree (style)	251
\Acrlong	66, 67, 196	alttreegroup (style)	253
\acrlong	22, 66, 67, 196	alttreehypergroup (style)	253
\ACRlongpl	196	amsgen package	89, 129
\Acrlongpl	67, 196	\andname	100
\acrlongpl	67, 196	array package	77, 80, 229, 241
\acrnameformat	63, 63, 197, 204	B	
acronym (option)	92	babel package	
\acronymfont	27, 60,	12,	
63, 65, 66, 197, 201, 206, 208, 209		13, 15, 21, 28, 98–100, 111, 269	
acronymlists (option)	93	C	
\acronymname	14, 99	counter (key)	
\acronymtype	23, 25, 31, 59, 59, 92, 194	counter (option)	
\acrpluralsuffix	68, 195	\CustomAcronymFields	
\ACRshort	66, 195	67, 68, 69, 213	
\Acrshort	66, 67, 195	\CustomNewAcronymDef	
		213	

D	
<code>\DeclareAcronymList</code>	92
<code>\DefaultNewAcronymDef</code>	198 , 266
<code>\defglssdisplay</code>	46 , 62 , 67–69 , 115 , 116 , 118 , 128 , 129
<code>\defglssdisplayfirst</code>	46 , 62 , 67–69 , 115 , 116 , 118 , 128 , 129
<code>\DefineAcronymSynonyms</code>	215
<code>\delimN</code>	104 , 191
<code>\delimR</code>	104 , 191
<code>description</code> (environment)	74 , 75 , 83 , 221
<code>description</code> (key)	116
<code>description</code> (option)	96
<code>descriptionaccess</code> (key)	255
<code>\DescriptionDUANewAcronymDef</code> ...	202
<code>\DescriptionFootnoteNewAcronymDef</code>	199 , 267
<code>\descriptionname</code>	14 , 99
<code>\DescriptionNewAcronymDef</code> ..	204 , 267
<code>descriptionplural</code> (key)	117
<code>descriptionpluralaccess</code> (key) ...	255
<code>dua</code> (option)	96
<code>\DUANewAcronymDef</code>	211
E	
<code>\emph</code>	38
<code>\entryname</code>	14 , 99
environments:	
<code>description</code>	74 , 75 , 83 , 221
<code>equation</code>	8
<code>itemize</code>	85
<code>longtable</code>	75–78 , 86 , 91 , 217 , 224–234
<code>supertabular</code>	78 , 80 , 91 , 217 , 234–246
<code>theglossary</code>	82 , 82 , 83 , 85 , 104 , 188 , 188 , 189 , 191 , 248–251
<code>theindex</code>	246
<code>equation</code> (environment)	8
F	
file types	
<code>.alg</code>	17
<code>.aux</code>	17 , 55 , 186
<code>.glg</code>	17–19 , 22
<code>.glo</code>	18 , 19 , 28 , 124
<code>.gls</code>	18 , 19 , 29
<code>.ist</code>	18 , 20 , 27 , 29 , 174 , 180
<code>.log</code>	22
<code>.tex</code>	18
<code>.toc</code>	106
<code>.xdy</code>	18 , 20 , 27 , 29 , 54 , 102
<code>\findrootlanguage</code>	110
<code>first</code> (key)	117
<code>first use</code>	21 , 28 , 45–47 , 59 , 60 , 62–65 , 71
<code>flag</code>	37 , 38 , 40 , 41 , 49
<code>text</code>	31 , 50
<code>firstaccess</code> (key)	254
<code>\firstacronymfont</code>	60 , 197
<code>firstplural</code> (key)	117
<code>firstpluralaccess</code> (key)	254
<code>flowfram</code> package	78
<code>fmtcount</code> package	56
<code>footnote</code> (option)	95
<code>\FootnoteNewAcronymDef</code>	206 , 268
<code>\forall glossaries</code>	113
<code>\forall gloss entries</code>	113
<code>\forall gloss entries</code>	113
G	
<code>german</code> package	13
<code>glossaries</code> package	
.....	111 , 112 , 175 , 217 , 221 , 254
<code>glossaries-accsupp</code> package	12 , 87 , 124 , 253
<code>glossaries-babel</code> package	15 , 28
<code>glossaries-polyglossia</code> package	15 , 28
<code>\GlossariesWarning</code>	97
<code>\GlossariesWarningNoLine</code>	97
<code>\glossary</code>	114 , 180 , 182 , 190
glossary keys:	
<code>access</code>	254
<code>counter</code>	118
<code>description</code>	116
<code>descriptionaccess</code>	255
<code>descriptionplural</code>	117
<code>descriptionpluralaccess</code>	255
<code>first</code>	117
<code>firstaccess</code>	254
<code>firstplural</code>	117
<code>firstpluralaccess</code>	254
<code>name</code>	116
<code>nonumberlist</code>	118
<code>parent</code>	118
<code>plural</code>	117
<code>pluralaccess</code>	254
<code>see</code>	118
<code>sort</code>	117
<code>symbol</code>	118
<code>symbolaccess</code>	254
<code>symbolplural</code>	118

symbolpluralaccess	255	long4col	72, 76, 227, 228
text	117	long4colborder	76, 77, 228
textaccess	254	long4colheader	76, 77, 227
type	118	long4colheaderborder	76, 77, 228
user1	118	longborder	76, 225
user2	119	longheader	76, 83, 225
user3	119	longheaderborder	76, 225
user4	119	longragged	77, 230, 231
user5	119	longragged3col	77, 78, 231, 232
user6	119	longragged3colborder	78, 232
glossary package	4, 21, 70, 194	longragged3colheader	78, 232
glossary styles:		longragged3colheaderborder	78, 232
altlist	75, 222, 223	longraggedborder	77, 230
altlistgroup	75, 223	longraggedheader	77, 231
altlisthypergroup	75, 223	longraggedheaderborder	77, 231
altlong4col	72, 76, 228, 229, 233	sublistdotted	224
altlong4colborder	77, 229	super	78, 79, 235, 236, 242
altlong4colheader	77, 228	super3col	79, 236, 237
altlong4colheaderborder	77, 229	super3colborder	79, 237
altlongragged4col	78, 233	super3colheader	79, 237
altlongragged4colborder	78, 233	super3colheaderborder	79, 237
altlongragged4colheader	78, 233	super4col	72, 79, 238, 239
altlongragged4colheaderborder		super4colborder	79, 239
	78, 234	super4colheader	79, 238
altsuper4col	72, 79, 239, 240, 244	super4colheaderborder	79, 239
altsuper4colborder	79, 240	superborder	78, 235
altsuper4colheader	79, 239	superheader	79, 236
altsuper4colheaderborder	79, 240	superheaderborder	79, 236
altsuperragged4col	81, 244, 245	superragged	80, 241, 242
altsuperragged4colborder	81, 245	superragged3col	80, 81, 242–244
altsuperragged4colheader	81, 245	superragged3colborder	80, 243
altsuperragged4colheaderborder		superragged3colheader	80, 243
	81, 245	superragged3colheaderborder	
alttree	81, 82, 250, 251, 253		80, 244
alttreegroup	82, 253	superraggedborder	80, 242
alttreehypergroup	82, 253	superraggedheader	80, 242
index	62, 63, 81, 246–248	superraggedheaderborder	80, 242
indexgroup	81, 247	superraggedright3colheaderborder	
indexhypergroup	81, 247		244
list	26, 74, 75, 83, 84, 86, 90, 221–223	tree	81, 82, 248, 249, 251
listdotted	74, 75, 223, 224	treegroup	81, 248, 249
listgroup	74, 222	treehypergroup	81, 249
listhypergroup	74, 75, 81–83, 222	treenoname	82, 249, 250
long	74, 76, 224–226, 230	treenonamegroup	82, 250
long3col	73, 76, 226	treenonamehypergroup	82, 250
long3colborder	73, 76, 226	glossary-hypernav package	174
long3colheader	73, 76, 226	glossary-list package	26, 54, 74, 90, 91, 221
long3colheaderborder	73, 76, 227		

glossary-long package	25, 106
. . . 26, 54, 75, 77, 91, 224, 233–235	
glossary-longragged package	54, 174
. . . 77, 229	
glossary-super package	102, 176
. . . 26,	
54, 78, 80, 91, 224, 234, 240, 244	
glossary-superragged package	94, 115
. . . 80, 240	
glossary-tree package	23, 92
. . . 26, 54, 81, 91, 246	
\glossaryentryfield	91
. 84, 85, 86, 118, 189, 191	
\glossaryentrynumber	43, 155
. 188	
\glossaryentrynumbers	43, 154
. 84, 90, 103, 186, 187	
\glossaryheader	43, 66, 154, 155, 196
. . . 75, 83, 85, 189, 191	
\glossarymark	156
. 24, 53, 105	
\glossaryname	156
. 14, 99, 100	
\glossarypostamble	155, 156
. 54, 104, 191	
\glossarypreamble	259
. 53, 104, 191	
\glossarysection	259
. 89, 104, 114	
\glossarystyle	259
. 26, 53, 72, 74, 191, 217	
\glossarysubentryfield	259
. 84, 86	
\GLS	259
. 40, 141	
\Gls	259
. 12, 40, 67, 70, 140, 143, 217	
\gls	259
. 31, 37, 39, 41, 45–48,	
67, 70, 71, 89, 118, 127, 128,	
139, 141, 142, 146, 148, 149,	
151, 152, 154, 155, 157, 158,	
160, 161, 163, 164, 166, 167	
\gls@checkisacronymlist	93
\gls@codepage	97
\gls@docclearpage	106
\gls@hypergrouprrun	220
\gls@level	119
\gls@suffixF	103
\gls@suffixFF	103
\glsaccessdisplay	259
\glsacccsupp	257
\glsadd	47, 127, 173, 190
\glsadd options	
counter	173
format	173, 191
\glsaddall	8, 47, 127, 173
\glsaddall options	
types	47, 173
\GlsAddSortRule	109
\GlsAddXdyAlphabet	108
\GlsAddXdyAttribute	38, 56, 107
\GlsAddXdyLocation	57, 108
\GlsAddXdyStyle	110
\glsautoprefix	25, 90
\glsclearpage	25, 106
\glsclosebrace	54, 174
\glscompositor	102, 176
\glscounter	94, 115
\glsdefaulttype	23, 92
\glsdefmain	91
\GLSdesc	43, 155
\Glsdesc	43, 154
\glsdesc	43, 66, 154, 155, 196
\GLSdescplural	156
\Glsdescplural	156
\glsdescplural	155, 156
\glsdescriptionaccessdisplay	259
\glsdescriptionpluralaccessdisplay	259
\glsdescwidth	
. 72, 76–80, 224, 230, 234, 241	
\glsdisablehyper	47, 138
\glsdisp	38, 40, 45–47, 145
\glsdisplay	31,
40, 41, 45, 69, 94, 116, 118, 128, 139	
\glsdisplayfirst	31, 40, 41,
45, 69, 116, 118, 128, 129, 139	
\glsdoifexists	114
\glsdoifnoexists	114
\glsenablehyper	47, 138
\glsentryaccess	256
\Glsentrydesc	50, 169
\glsentrydesc	50, 94, 169
\glsentrydescaccess	257
\Glsentrydescplural	50, 170
\glsentrydescplural	50, 170
\glsentrydescpluralaccess	257
\Glsentryfirst	50, 171
\glsentryfirst	49, 171
\glsentryfirstaccess	256
\Glsentryfirstplural	50, 171
\glsentryfirstplural	50, 171
\glsentryfirstpluralaccess	256
\Glsentryname	49, 169
\glsentryname	49, 52, 169
\Glsentryplural	49, 170
\glsentryplural	49, 170
\glsentrypluralaccess	256
\glsentrysort	171
\Glsentrysymbol	50, 170
\glsentrysymbol	50, 170
\glsentrysymbolaccess	257
\Glsentrysymbolplural	51, 171

<code>\glsentrysymbolplural</code>	50, 170	<code>\glslongaccesskey</code>	269
<code>\glsentrysymbolpluralaccess</code>	257	<code>\glslongkey</code>	61, 195
<code>\Glsentrytext</code>	49, 170	<code>\glslongpluralaccesskey</code>	269
<code>\glsentrytext</code>	49, 170	<code>\glslongpluralkey</code>	61, 195
<code>\glsentrytextaccess</code>	256	<code>\glslongtok</code>	68, 197
<code>\glsentrytype</code>	171	<code>\GLSname</code>	42, 153
<code>\Glsentryuseri</code>	51, 171	<code>\Glsname</code>	42, 153
<code>\glsentryuseri</code>	51, 171	<code>\glsname</code>	42, 152, 153
<code>\Glsentryuserii</code>	51, 172	<code>\glsnameaccessdisplay</code>	257
<code>\glsentryuserii</code>	51, 172	<code>\glsnamefont</code>	54, 191
<code>\Glsentryuseriii</code>	51, 172	<code>\glsnavhyperlink</code>	219
<code>\glsentryuseriii</code>	51, 69, 172	<code>\glsnavhypertarget</code>	83, 219
<code>\Glsentryuseriv</code>	51, 172	<code>\glsnavigation</code>	220
<code>\glsentryuseriv</code>	51, 172	<code>\glsnonextpages</code>	188
<code>\Glsentryuserv</code>	51, 172	<code>\glsnoxindywarning</code>	106
<code>\glsentryuserv</code>	51, 172	<code>\glsnumberformat</code>	103
<code>\Glsentryuservi</code>	51, 172	<code>\glsnumbersgroupname</code>	14, 99, 174, 190
<code>\glsentryuservi</code>	51, 172	<code>\glsopenbrace</code>	54, 174
<code>\GLSfirst</code>	41, 149	<code>\glsorder</code>	96
<code>\Glsfirst</code>	41, 148, 149	<code>\glspagelistwidth</code>	
<code>\glsfirst</code>	41, 66, 148, 196		72, 76–80, 224, 230, 235, 241
<code>\glsfirstaccessdisplay</code>	258	<code>\glspar</code>	30, 101
<code>\GLSfirstplural</code>	42, 152	<code>\GLSpl</code>	31, 40, 144
<code>\Glsfirstplural</code>	42, 151	<code>\Glspl</code>	31, 40, 67, 143, 217
<code>\glsfirstplural</code>	42, 151, 152	<code>\glspl</code>	31, 37,
<code>\glsfirstpluralaccessdisplay</code>	258		40, 45–47, 67, 71, 127, 128, 142–144
<code>\glsgetgrouplabel</code>	190	<code>\GLSplural</code>	41, 150
<code>\glsgetgrouptitle</code>	83, 174, 190	<code>\Glsplural</code>	41, 150
<code>\glsgroupheading</code>	83, 85, 190, 191	<code>\glsplural</code>	41, 149, 150
<code>\glsgroupskip</code>	7, 74, 83, 85, 189, 191, 221	<code>\glspluralaccessdisplay</code>	258
<code>\glshyperlink</code>	49, 52, 173	<code>\glspluralsuffix</code>	31, 32, 99, 117
<code>\glshypernavsep</code>	74, 221	<code>\glspostdescription</code>	74, 101
<code>\glshypernumber</code>	103, 191	<code>\glsquote</code>	55, 174
<code>\glsIfListOfAcronyms</code>	93	<code>\glsreset</code>	71, 126
<code>\glskeylisttok</code>	197	<code>\glsresetall</code>	71, 126
<code>\glslabel</code>	45	<code>\glsresetentrylist</code>	188
<code>\glslabeltok</code>	68, 197	<code>\glssee</code>	4, 38, 48, 184
<code>\glslink</code>	37, 40, 45,	<code>\glsseeformat</code>	177
	47, 127, 128, 130, 139, 173, 190, 191	<code>\glsseeitem</code>	185
<code>\glslink options</code>		<code>\glsseelastsep</code>	185
counter	39, 129, 139	<code>\glsseelist</code>	184
format	38, 56, 129, 139, 191	<code>\glsseesep</code>	185
hyper	39, 47, 129, 139	<code>\glsSetAlphaCompositor</code>	30, 103
<code>\glslink*</code>	39	<code>\glsSetCompositor</code>	29, 102
<code>\glslistdottedwidth</code>	75, 224	<code>\glsSetSuffixF</code>	36, 103
<code>\glslocalreset</code>	71, 126	<code>\glsSetSuffixFF</code>	36, 103
<code>\glslocalresetall</code>	71, 127	<code>\glssettocitle</code>	99
<code>\glslocalunset</code>	71, 126	<code>\glssetwidest</code>	82, 250
<code>\glslocalunsetall</code>	71, 127	<code>\GlsSetXdyCodePage</code>	17, 55, 112

<code>\GlsSetXdyFirstLetterAfterDigits</code>	174	
<code>\GlsSetXdyLanguage</code>	17 , 55 , 111	
<code>\GlsSetXdyLocationClassOrder</code>	57 , 109	
<code>\GlsSetXdyMinRangeLength</code>	36 , 57 , 175	
<code>\GlsSetXdyStyles</code>	110	
<code>\glsshortaccesskey</code>	269	
<code>\glsshortkey</code>	61 , 195	
<code>\glsshortpluralaccesskey</code>	269	
<code>\glsshortpluralkey</code>	61 , 195	
<code>\glsshorttok</code>	68 , 197	
<code>\GLSsymbol</code>	43 , 158	
<code>\Glsymbol</code>	43 , 157	
<code>\glssymbol</code>	42 , 66 , 157 , 158	
<code>\glssymbolaccessdisplay</code>	258	
<code>\glssymbolnav</code>	221	
<code>\GLSsymbolplural</code>	159	
<code>\Glsymbolplural</code>	159	
<code>\glssymbolplural</code>	158 , 159	
<code>\glssymbolpluralaccessdisplay</code>	258	
<code>\glssymbolsgroupname</code>	14 , 99 , 174 , 190	
<code>\glstarget</code>	84 , 189	
<code>\GLStext</code>	41 , 147	
<code>\Glstext</code>	41 , 147	
<code>\glstext</code>	41 , 41 , 42 , 43 , 66 , 146 , 195	
<code>\glstextaccessdisplay</code>	257	
<code>\glstextformat</code>	37 , 46 , 128	
<code>\glstocfalse</code>	23	
<code>\glstoctrue</code>	23	
<code>\glstreeindent</code>	81 , 248–250	
<code>\glunset</code>	71 , 126	
<code>\glunsetall</code>	71 , 127	
<code>\GLSuseri</code>	43 , 161	
<code>\Glsuseri</code>	43 , 160	
<code>\gluseri</code>	43 , 160 , 161	
<code>\GLSuserii</code>	44 , 162	
<code>\Glsuserii</code>	44 , 162	
<code>\gluserii</code>	43 , 161 , 162	
<code>\GLSuseriii</code>	44 , 164	
<code>\Glsuseriii</code>	44 , 163	
<code>\gluseriii</code>	44 , 163 , 164	
<code>\GLSuseriv</code>	44 , 165	
<code>\Glsuseriv</code>	44 , 165	
<code>\gluseriv</code>	44 , 164 , 165	
<code>\GLSuserv</code>	44 , 167	
<code>\Glsuserv</code>	44 , 166	
<code>\gluserv</code>	44 , 166 , 167	
<code>\GLSuservi</code>	45 , 168	
<code>\Glsuservi</code>	44 , 168	
<code>\gluservi</code>	44 , 167 , 168	
		H
html package	47	
<code>\hyperbf</code>	39 , 193	
<code>\hyperemph</code>	39 , 193	
hyperfirst (option)	95	
<code>\hyperit</code>	39 , 193	
<code>\hyperlink</code>	38 , 47 , 138	
<code>\hypermd</code>	39 , 193	
<code>\hyperpage</code>	38 , 191	
hyperref package	36–38 , 47 , 69 , 84 , 185 , 191	
<code>\hyperm</code>	39 , 56 , 193	
<code>\hypersc</code>	39 , 193	
<code>\hypersf</code>	39 , 193	
<code>\hypersl</code>	39 , 193	
<code>\hypertarget</code>	47 , 138	
<code>\hypertt</code>	39 , 193	
<code>\hyperup</code>	39 , 193	
		I
<code>\if@glisacronymlist</code>	93	
<code>\ifglossaryexists</code>	113	
<code>\ifglentryexists</code>	113	
<code>\ifglused</code>	72 , 114 , 126	
<code>\ifglxindy</code>	96	
index (style)	246	
indexgroup (style)	247	
indexhypergroup (style)	247	
<code>\indexspace</code>	81	
inputenc package	10 , 12 , 32 , 55 , 88	
<code>\inputencodingname</code>	55 , 97	
<code>\istfilename</code>	102	
<code>\item</code>	191 , 221 , 246 , 247	
itemize (environment)	85	
		J
<code>\jobname</code>	29	
		L
<code>\languagename</code>	110	
link text	37 , 39 , 45 , 46 , 128	
list (style)	221	
listdotted (style)	223	
listgroup (style)	222	
listhypergroup (style)	222	
<code>\loadglsentries</code>	35 , 92 , 127	
location list	see number list	
long (style)	224	
long3col (style)	226	
long3colborder (style)	226	
long3colheader (style)	226	

<code>long3colheaderborder</code> (style)	227	<code>\newglossaryentry</code>	7 , 30 , 37 , 40 , 60 , 61 , 68 , 94 , 119 , 127 , 128 , 194
<code>long4col</code> (style)	227	<code>\newglossaryentry</code> options	
<code>long4colborder</code> (style)	228	access	87 , 255 , 256
<code>long4colheader</code> (style)	227	counter	118
<code>long4colheaderborder</code> (style)	228	description	26 , 27 , 30 , 31 , 43 , 45 , 61 , 63 , 68 , 69 , 87 , 94 , 96 , 116 , 117 , 119 , 120 , 128 , 154 , 169 , 195 , 206 , 255
<code>longborder</code> (style)	225	descriptionaccess	87 , 257 , 259
<code>longheader</code> (style)	225	descriptionplural	31 , 45 , 87 , 155 , 255
<code>longheaderborder</code> (style)	225	descriptionpluralaccess	87 , 257 , 259
<code>longragged</code> (style)	230	first	31 , 37 , 39–41 , 45 , 50 , 61 , 68 , 69 , 71 , 87 , 117 , 122 , 128 , 139 , 148 , 171 , 204 , 209 , 254
<code>longragged3col</code> (style)	231	firstaccess	87 , 256 , 258
<code>longragged3colborder</code> (style)	232	firstplural 31 , 33 , 40 , 42 , 45 , 50 , 68 , 69 , 87 , 117 , 122 , 128 , 151 , 171 , 254	
<code>longragged3colheader</code> (style)	232	firstpluralaccess	87 , 256 , 258
<code>longragged3colheaderborder</code> (style)	232	format	39 , 175
<code>longraggedborder</code> (style)	230	name	26 , 27 , 30 , 31 , 34 , 42 , 49 , 63 , 68 , 83 , 87 , 94 , 95 , 116 , 117 , 119 , 120 , 152 , 169 , 254
<code>longraggedheader</code> (style)	231	nonumberlist	31 , 118
<code>longraggedheaderborder</code> (style)	231	parent	30 , 33 , 118 , 119
<code>longtable</code> (environment)	75–78 , 86 , 91 , 217 , 224–234	plural	31–34 , 40 , 41 , 45 , 49 , 68 , 69 , 87 , 117 , 122 , 128 , 149 , 254
<code>longtable</code> package	26 , 76 , 224 , 229	pluralaccess	87 , 256 , 258
M			
<code>\makefirsttuc</code>	65 , 87 , 217	see	4 , 32 , 38 , 48 , 118
<code>makeglossaries</code>	4 , 6–12 , 15–19 , 21 , 22 , 27–29 , 48 , 52 , 55 , 56 , 58 , 96 , 102 , 110–112 , 115 , 186	sort	22 , 27 , 31 , 32 , 34 , 63 , 83 , 94 , 117 , 171 , 189 , 190
<code>\makeglossaries</code>	15 , 22 , 29 , 36 , 37 , 52 , 56–59 , 98 , 102 , 103 , 114 , 116 , 181 , 182	symbol	26 , 27 , 31 , 42 , 45 , 46 , 61–63 , 68 , 69 , 87 , 94 , 95 , 116 , 118 , 128 , 157 , 170 , 200– 202 , 204 , 209 , 227 , 238 , 254 , 255
<code>\makeglossary</code>	182	symbolaccess	87 , 257 , 258
<code>makeindex</code>	4 , 6–12 , 15–22 , 27 , 29 , 31 , 32 , 36–38 , 52 , 58 , 73 , 81 , 83 , 94 , 96 , 99 , 102 , 104 , 114–117 , 125 , 131 , 133 , 174 , 177 , 179 , 180 , 183 , 189 , 190	symbolplural	31 , 45 , 87 , 158 , 255
delim_n	104	symbolpluralaccess	87 , 257 , 258
delim_r	104	text	27 , 31 , 37 , 39–41 , 45 , 49 , 61 , 68 , 69 , 71 , 87 , 117 , 128 , 139 , 146 , 170 , 200 , 204 , 254
page_compositor	102	textaccess	87 , 256 , 257
special characters	132 , 174	type	31 , 35 , 92 , 118 , 127 , 171
<code>\MakeUppercase</code>	88	user1	3 , 31 , 43 , 69 , 86 , 160 , 171 , 255
memoir class	182	user2	69 , 86 , 161 , 172
<code>mfirsttuc</code> package	87	user3	69 , 163 , 172
N		user4	69 , 164 , 172
name (key)	116	user5	166 , 172
<code>\newacronym</code>	7 , 23 , 27 , 59 , 60 , 63 , 67 , 68 , 70 , 96 , 127 , 128 , 194 , 195	user6	3 , 31 , 86 , 167 , 172 , 255
<code>\newacronymhook</code>	197	<code>\newglossarystyle</code>	74 , 82 , 191
<code>\newglossary</code>	18 , 19 , 23 , 58 , 93 , 115 , 116 , 180 , 181 , 187		

`\newline` [30](#), [72](#)
`ngerman package` [13](#), [55](#), [111](#)
`\nohyperpage` [36](#)
`\noist` [10](#), [29](#), [36](#), [37](#), [55–58](#), [180](#)
`nolist (option)` [91](#)
`nolong (option)` [91](#)
`nonumberlist (key)` [118](#)
`nonumberlist (option)` [90](#)
`\nopostdesc` [30](#), [33](#), [34](#), [74](#), [101](#)
`nostyles (option)` [91](#)
`nosuper (option)` [91](#)
`notree (option)` [91](#)
`number list` [8](#), [16](#), [26](#), [29–31](#),
[33](#), [34](#), [36](#), [37](#), [49](#), [57](#), [58](#), [74–80](#), [85](#)
`numberedsection (option)` [90](#)
`numberline (option)` [89](#)

O

`\oldacronym` [70](#), [70](#), [194](#)
`order (option)` [96](#)

P

package options:

`acronym` [14](#), [18](#), [19](#), [23](#),
[25](#), [31](#), [35](#), [36](#), [59](#), [92](#), [99](#), [187](#), [194](#)
`acronym` [92](#)
`acronymlists` [24](#), [59](#)
`acronymlists` [93](#)
`counter` [26](#), [29](#), [36](#), [94](#)
`counter` [94](#)
`description` [27](#), [60–65](#), [70](#), [204](#)
`description` [96](#)
`dua` [27](#), [60–63](#), [65](#), [204](#)
`dua` [96](#)
`footnote` [27](#), [59–65](#), [70](#), [139](#),
[141–146](#), [201](#), [204](#), [206](#), [260–265](#)
`footnote` [95](#)
`hyperfirst` [28](#)
`false` [21](#), [47](#), [139](#), [141–146](#)
`true` [28](#)
`hyperfirst` [95](#)
`makeindex` [27](#)
`nolist` [26](#), [74](#), [217](#)
`nolist` [91](#)
`nolong` [26](#), [72](#), [75](#), [217](#), [224](#)
`nolong` [91](#)
`nomain` [23](#), [59](#), [92](#)
`nonumberlist` [26](#), [36](#), [84](#), [90](#)
`nonumberlist` [90](#)

`nostyles` [26](#), [72](#), [74](#), [75](#), [78](#), [81](#)
`nostyles` [91](#)
`nosuper` [26](#), [72](#), [78](#), [217](#)
`nosuper` [91](#)
`notree` [26](#), [81](#), [217](#)
`notree` [91](#)
`nowarn` [23](#)
`numberedsection` [25](#), [53](#)
`autolabel` [25](#)
`false` [25](#)
`nolabel` [25](#)
`numberedsection` [90](#)
`numberline` [23](#), [89](#)
`numberline` [89](#)
`order` [28](#)
`letter` [10](#), [17](#), [28](#)
`word` [10](#), [17](#), [28](#)
`order` [96](#)
`sanitize` [26](#), [62](#),
[65](#), [67](#), [94](#), [95](#), [116](#), [117](#), [169](#), [170](#)
`description` [62](#), [64](#)
`none` [27](#)
`symbol` [62](#), [63](#), [65](#)
`sanitize` [95](#)
`section` [24](#), [53](#), [89](#), [105](#)
`section` [89](#)
`shortcuts` [27](#), [60](#), [67](#), [68](#)
`shotcuts` [96](#)
`smallcaps` [20](#), [27](#), [59–62](#), [64](#), [65](#)
`smallcaps` [96](#)
`smaller` [21](#), [27](#), [60–62](#), [65](#)
`smaller` [96](#)
`style` [26](#), [53](#), [72](#), [77](#), [80](#), [90](#), [217](#)
`style` [90](#)
`toc` [23](#), [52](#), [89](#)
`true` [89](#)
`toc` [89](#)
`translate` [28](#), [95](#)
`false` [15](#), [28](#)
`true` [28](#)
`translate` [95](#)
`xindy` [11](#), [12](#),
[16–18](#), [27](#), [28](#), [30](#), [54](#), [58](#), [97](#)
`\pagelistname` [14](#), [99](#)
`parent (key)` [118](#)
`\phantomsection` [104](#), [105](#)
`plural (key)` [117](#)
`pluralaccess (key)` [254](#)
`pod2man` [17](#)

polyglossia package	12, 15, 28, 100	\setStyleFile	18, 19, 29, 102
\printglossaries	22, 29, 52, 91, 104, 114, 116, 182, 185, 187, 219	shotcuts (option)	96
\printglossary	22, 26, 29, 52, 72, 104, 114, 182, 185, 187, 219	smallcaps (option)	96
\printglossary options		\smaller	65
nonumberlist	53, 188	smaller (option)	96
numberedsection	53, 188	\SmallNewAcronymDef	208, 268
style	26, 53, 72, 187	sort (key)	117
title	52, 187	style (option)	90
toctitle	52, 187	\subitem	247
type	52, 92, 185, 187	sublistdotted (style)	224
\protect	94	\subsubitem	247
R			
relsize package	27, 61, 65	super (style)	235
\roman	175	super3col (style)	236
\rootlanguage	111	super3colborder (style)	237
S			
sanitize (option)	95	super3colheader (style)	237
section (option)	89	super3colheaderborder (style)	237
see (key)	118	super4col (style)	238
\seename	49, 99	super4colborder (style)	239
\SetAcronymLists	93	super4colheader (style)	238
\SetAcronymStyle	92, 212	super4colheaderborder (style)	239
\setAlphaCompositor	57	superborder (style)	235
\setCompositor	57	superheader (style)	236
\SetCustomDisplayStyle	68, 69, 213	superheaderborder (style)	236
\SetCustomStyle	68, 214	superragged (style)	241
\SetDefaultAcronymDisplayStyle	197	superragged3col (style)	242
\SetDefaultAcronymStyle	198	superragged3colborder (style)	243
\SetDescriptionAcronymDisplayStyle	204	superragged3colheader (style)	243
.....	204	superraggedborder (style)	242
\SetDescriptionAcronymStyle	204	superraggedheader (style)	242
\SetDescriptionDUAAcronymDisplayStyle	201	superraggedheaderborder (style)	242
.....	202	superraggedright3colheaderborder (style)	244
\SetDescriptionFootnoteAcronymDisplayStyle	199	supertabular (environment)	78, 80, 91, 217, 234–246
.....	200	supertabular package	26, 78, 80, 91, 217, 234, 241
\SetDescriptionFootnoteAcronymStyle	200	symbol (key)	118
\SetDUADisplayStyle	210	symbolaccess (key)	254
\SetDUASyle	211	\symbolname	14, 99
\setentrycounter	190	symbolplural (key)	118
\SetFootnoteAcronymDisplayStyle	206	symbolpluralaccess (key)	255
\SetFootnoteAcronymStyle	206	T	
\setglossarysection	24, 105	text (key)	117
\SetSmallAcronymDisplayStyle	208	textaccess (key)	254
\SetSmallAcronymStyle	209	\textbf	38
		\textrm	56
		\textsc	64
		\textsmaller	27, 65

theglossary (environment)	82 , 82 , 83 , 85 , 104 , 188 , 188 , 189 , 191 , 248–251	user4 (key)	119
theindex (environment)	246	user5 (key)	119
\thepage	56	user6 (key)	119
toc (option)	89		
\translate	100	W	
translate (option)	95	\warn@nomakeglossaries	181
translator package	12 , 13 , 15 , 22 , 28 , 98 , 100 , 185 , 269 , 278–281	\warn@noprintglossary	185
tree (style)	248	\writeist	102 , 107 , 108 , 175
treegroup (style)	248		
treehypergroup (style)	249	X	
treenoname (style)	249	\xglscacsupp	257
treenonamegroup (style)	250	xindy	4 , 10– 12 , 15–22 , 27–32 , 36–39 , 52 , 54– 58 , 81 , 83 , 94 , 96 , 97 , 102 , 103 , 106 , 108–112 , 124 , 125 , 136 , 137 , 174 , 175 , 177 , 183 , 186 , 189
treenonamehypergroup (style)	250	xkeyval package	6 , 20
type (key)	118	\xmakefirstuc	88 , 219
		\xspace	70
U		xspace package	70 , 89 , 194
user1 (key)	118		
user2 (key)	119		
user3 (key)	119		